VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Lokalizační systém pro mobilní robotické zařízení

Localisation System for Mobile Robotic Device

2013 **Bc. Aleš Kurečka**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Aleš Kurečka**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Lokalizační systém pro mobilní robotické zařízení
Localisation System for Mobile Robotic Device

Zásady pro vypracování:

Práce se bude zabývat lokalizačním systémem pro mobilní robotické zařízení. Úkolem studenta je integrace modulu lokalizace pomocí úzlů bezdrátové sítě do stávajícího lokalizačního systému a celkové zvýšení robustnosti lokalizačního systému.

1. Průzkumné vozidlo a ovládací panel.
2. Možnosti lokalizace. Lokalizace pomocí bezdrátové sítě.
3. Návrh zdokonalení stávajících lokalizačních systémů.
4. Návrh a realizace lokalizačních algoritmů.
5. Platforma 32b s RTOS (např. ARM Freescale iMX31 LiteKit, Timesys Linux).
6. Integrace do stávajícího systému.
7. Testování řešení.
8. Dokumentace řešení.
9. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] *Timesys - Embedded Linux software* [online]. c2009 [cit. 2009-11-11]. Dostupné z: http://linuxlink.timesys.com/.
[2] *Logic : A Product Development and Manufacturing Company* [online]. c2009 [cit. 2009-11-11]. Dostupné z: http://www.logicpd.com/.
[3] LIU, Yunhao a Zheng YANG. *Location, Localization and Localizability : Location-awareness Technology for Wireless Networks.* New York : Springer, 2011. 154 s. ISBN 978-1-4419-7370-2, DOI 10.1007/978-1-4419-7371-9.
[4] ADDESSO, Paolo, Luigi BRUNO a Rocco RESTAINO. Adaptive localization techniques in WiFi environments. In *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium* . New York : IEEE Press, 2010. s. 289-294. ISBN 978-1-4244-6855-3 , DOI:10.1109/ISWPC.2010.5483731.
[5] CONNECTBLUE AB. *OWSPA311g : Electrical and mechanical datasheet.* Sweden : [s.n.], 2008. 46 s.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jiří Kotzian, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal, a nejsou mi známy žádné okolnosti, které by mohly vést k pochybnostem o mojí práci.

.....

Aleš Kurečka

Datum odevzdání diplomové práce: 7.5.2013

Poděkování

Rád bych na prvním místě poděkoval vedoucímu mé bakalářské práce, panu Ing. Jiřímu Kotzianovi, Ph.D. za obrovskou trpělivost, přínosné rady, připomínky a celkové směřování při tvorbě této práce. Bez jeho podpory by tato práce nevznikla.

Dále panu Ing. Jaromíru Konečnému za úvod do problematiky cílové platformy a seznámení s aplikačním vybavením vozidla a za podporu při praktické části této práce.

Abstrakt

Práce se zabývá integrací lokalizačního systému využívající senzorická data existujícího autonomního mobilního průzkumného vozidla.

Vozidlo je vybaveno mnoha senzory, které jsou pro účely lokalizace vhodné, jako je laserový senzor vzdáleností, ultrazvukové dálkoměry, GPS a v neposlední řadě i WiFi modul. Vozidlo aktuálně oznamuje svou polohu pomocí GPS, která je však nepřesná a v budovách obvykle nepoužitelná. Komplexní lokalizační systém, který by poskytoval polohu vozidla využitelnou k orientaci v prostoru a autonomnímu řízení však dosud chybí.

Výstupem práce je pravděpodobnostní lokalizační systém založený na metodě Monte-Carlo, který agreguje data poskytovaná ze senzorů, z odometrie, modulů GPS, WiFi a vypočítává polohu vozidla v prostoru. Implementovaný modul tvoří modulární a dále rozšiřitelný robustní systém poskytující informaci o poloze i za předpokladu, že data z některých senzorů nejsou k dispozici.

Práce se také zabývá využitím přístupových bodů WiFi k lokalizačním účelům a implementuje ji jako jeden ze zdrojů lokalizace.

Klíčová slova

Pravděpodobnostní lokalizace, lokalizace Monte Carlo, částicový filtr, WiFi, RSSI, průzkumné vozidlo, Gaussův filtr, Timesys Linux, i.MX31.

Abstract

This thesis deals with the integration of localization module for existing autonomous mobile exploration vehicle.

The vehicle is equipped with many sensors that are suitable for localization purposes, such as laser distance sensor, ultrasonic rangefinders, GPS and also WiFi module. The vehicle currently announces its location using GPS, which is usually inaccurate and unusable in buildings. A comprehensive localization system suitable for autonomous control, is still missing.

The output of this work is probabilistic localization system based on Monte-Carlo method, which aggregates the data provided by the sensors, odometry, GPS modules, WiFi and calculates the vehicle's position in space. The developed module is modular and extensible robust system that provides location also in situation, that data from some sensors are not available.

The work also deals with the use of WiFi access points for localization purposes and implements it as a one of sources for localization.

Keywords

Probabilistic localization, Monte Carlo localization, particle filter, WiFi, RSSI, exploration vehicle, Gaussian filter, Timesys Linux, i.MX31.

Seznam použitých zkratek a symbolů

Zkratka/symbol	Význam
[-]	Označení bezrozměrné veličiny
s, ms	Jednotka času – sekunda, milisekunda
m, km	Jednotka vzdálenosti – metr, kilometr
dB	Jednotka logaritmické míry – decibel
dBm	Jednotka logaritmické míry vztažená na miliwatt – decibel miliwatt
dBi	Jednotka logaritmické míry používaná k vyjádření zisku izotropních antén – decibel izotropic
MHz, GHz	Jednotka frekvence – megahertz, gigahertz
UHF	Ultra high frequency – ultra krátké vlny, 300 MHz až 3 GHz
SNR	Signal to Noise Ratio – poměr výkonu signálu k šumu
RSSI	Received Signal Strength Indication – útlum přijatého signálu
AoA	Angle of Arrival – lokalizační metoda založená na měření úhlů
PoA	Phase of Arrival – lokalizační metoda založená na měření fázových zpoždění
ToA	Time of Arrival – lokalizační metoda založená na měření času
TDoA	Difference of Arrival – lokalizační metoda založená na měření časových rozdílů
AP	Access Point, přístupový bod WiFi
AT	Attention – soubor příkazů zavedený firmou Hayes
GPS	Global Positioning Systém – družicový systém pro určení polohy
MAC	Media Access Control – jedinečný identifikátor síťového zařízení
WiFi	soubor standardů IEEE 802.11 pro bezdrátové sítě LAN
C++	Vyšší programovací jazyk
OS	Operating System – operační systém
RTOS	Real-Time Operating System – operační systém reálného času
Ethernet	Komunikační sběrnice
i.MX31	Výkonný multimediální procesor od firmy Freescale s jádrem ARM
Kernel	Jádro, v této práci používané v souvislosti s Gaussovým filtrem a jádrem OS Linux
Mutex	Synchronizační prostředek limitující přístup jednotlivých procesů do společného prostředku
MCL	Monte-Carlo Localization – Monte Carlo lokalizace
RS232	Komunikační standard sériové linky
UART	Asynchronní komunikační sériové rozhraní
TFTP	Trivial File Transfer Protocol - zjednodušený protokol FTP pro přenos souborů
UML	Unified Modeling Language - grafický nástroj pro softwarové inženýrství

Obsah

1 Úvod

Problematika lokalizace mobilních robotů je fundamentální a zcela nepostradatelná pro autonomní plánování trasy a následné řízení.

Na základě výzkumné činnosti vznikl na katedře měřicí a řídicí techniky na VŠB-TUO prototyp robotického průzkumného vozidla. Vozidlo je pro potřebu orientace v terénu vybaveno množstvím senzorů. Mimo manuálního ovládání umožňuje i jistý autonomní pohyb na zadané místo pomocí odometrie a mapy s vyznačenými cestami[1]. Pro určení absolutní pozice v prostoru využívá lokalizaci pomocí GPS modulu. Lokalizace probíhá na základě multilaterace[2]. Předpokladem správné a přesné lokalizace v 2D[3] prostoru je přímá viditelnost s minimálně 3mi satelity tohoto systému a dostatečný signál. Civilní GPS používá frekvenci 1575,42 MHz, tedy pásmo UHF. Tímto je použití této technologie omezeno na již zmíněnou přímou viditelnost a prakticky vylučuje spolehlivé použití v hustě zastavěných oblastech nebo budovách [1].

Pro lokalizaci v těchto oblastech se nabízí použití GSM, nebo WiFi, popřípadě některých dalších technologií vysílajících signál. Výhoda GSM je její pokrytí, nicméně přesnost je pro účely vozidla nedostačující[4]. V obou případech můžeme s výhodou použít stávající infrastrukturu. Navíc při WiFi lokalizaci není nutné připojení a tedy často i znalost klíče pro použitou metodu šifrování, ale pouze znalost síly signálu a MAC adresy AP jako jedinečného identifikátoru.

Tato diplomová práce navazuje na mou bakalářskou práci „*Vývoj embedded modulu pro podporu lokalizace průzkumného vozidla* “ v rámci které vznikla aplikace, vypočítávající polohu na základě multilaterace z RSSI viditelných AP [2].

Cílem této práce je rozšířit vozidlo o jednotný lokalizační systém, který využije jednak proximitní senzory využité na vozidle, zejména pak laserový senzor vzdáleností SICK LMS 100, modul GPS a také informace z viditelných AP na okolní WiFi. Systém musí být robustní ve smyslu správné činnosti, i když data z některých senzorů nejsou použitelná[5] a dostatečně přesný pro účely autonomního řízení.

---

1 Odometrie je velmi nepřesná, takže je tato možnost použitelná jen na minimální vzdálenosti.

2 Mnohdy nesprávně nazýváno jako triangulace, která pracuje s trigonometrií, nikoliv vzdáleností .

3 Zde myšleno určení zeměpisné šířky a délky .

4 Řádově jednotky až desítky metrů.

5 Například GPS nepřijímá signál z žádného satelitu, vozidlo se nachází na místě, kde chybí mapa AP.

2 PRŮZKUMNÉ VOZIDLO A OVLÁDACÍ PANEL

Před předvedením konceptu řešení lokalizace bude objasněn aktuální stav průzkumného vozidla – tedy základu, na který je tato práce cílena. Bude rozebráno jeho vybavení, osazení senzory, vlastnosti a také řídicí modul, na kterém bude algoritmus lokalizace spuštěn. Průzkumné vozidlo je možné vzdáleně monitorovat a případně manuálně řídit ovládacím panelem, který bude také krátce zmíněn. Vzhled vozidla ilustruje [Obr. 1].

Obr. 1: Fotografie průzkumného vozidla [3]

2.1 MECHANICKÁ KONSTRUKCE

Vozidlo je sestaveno na robustním a objemném podvozku RC modelu Baja 5B SS[6] s rozměry 817x460 mm a rozvorem kol 570 mm. Původně osazený benzinový motor Fuelie 26SS byl nahrazen elektromotorem Maxon 304747. Tím došlo k uvolnění místa pro elektroniku a snížení celkové hmotnosti.

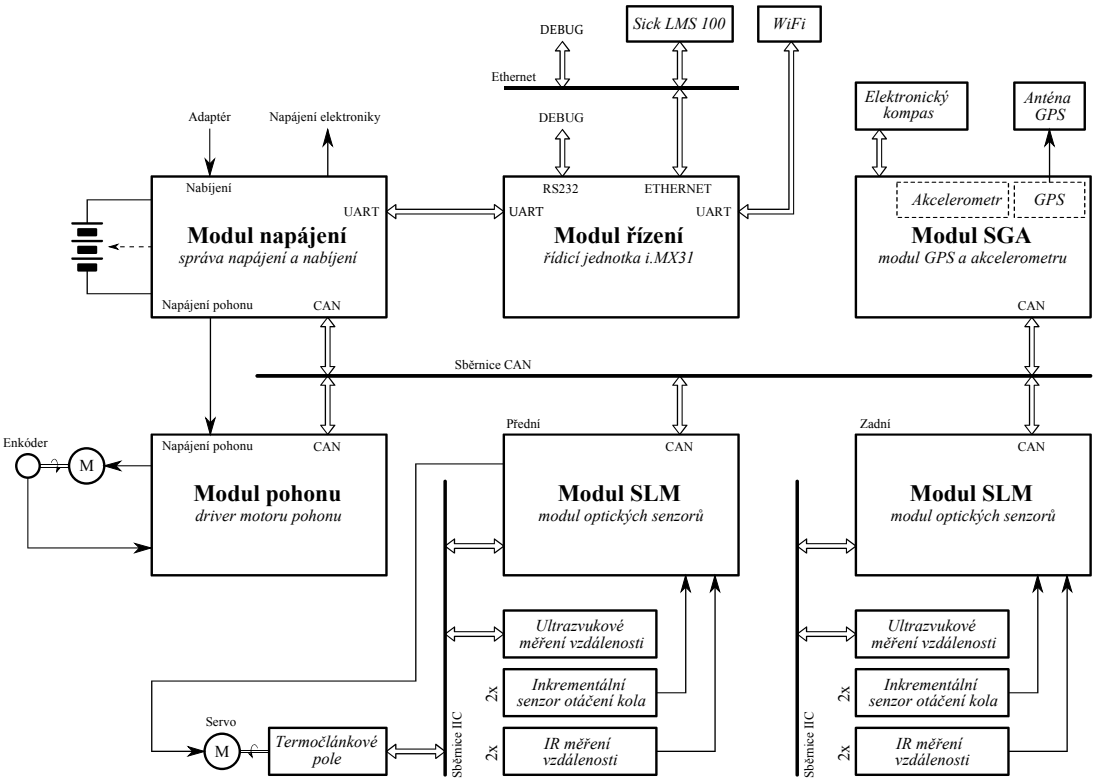
6 Stránka výrobce: http://www.hpieurope.com/kit-info.php?lang=en&partNo=10611.

Model je vybaven diferenciálem, převodovkou a tlumiči. Přední kola jsou vychylována servomotorem a jejich otáčení je snímáno inkrementálními senzory.

Elektronika včetně baterií a pohonu je nainstalována do rámu podvozku. Na střeše je upevněna pomocná konstrukce s laserovým senzorem SICK LMS 100, WiFi modulem a jeho anténou. Do předního i zadního nárazníku jsou instalovány ultrazvukové senzory vzdálenosti. Na skosených plochách nárazníků jsou navíc upevněny 4 analogové dálkoměry tak, aby snímali vzdálenost v různých směrech. Nad předním nárazníkem je umístěno termočlánkové pole rozmítané servomotorem. Vozidlo je dále vybaveno GPS modulem a elektronickým kompasem [3].

2.2 ELEKTRONIKA VOZIDLA

Elektronika průzkumného vozidla tvoří distribuovaný systém tvořený několika navzájem komunikujícími moduly. Její zjednodušené blokové schéma je na [Obr. 2]. Každý z těchto modulů je osazen vlastním mikrokontrolérem[7].



Obr. 2: Blokové schéma elektroniky vozidla

7 Jedná se o 16-ti bitové mikrokontroléry fy. Freescale, řady HCS12. Vyjímku tvoří řídicí jednotka osazená i.MX31.

2.2.1 Řídicí jednotka

Řídicí jednotka je tvořena vývojovým kitem i.MX31 LiteKit s výkonným 32bitovým mikroprocesorem [4]. Obsahuje kompletní softwarové vybavení pro autonomní i manuální řízení vozidla. Jednotka komunikuje s podřízenými moduly distribuovaného řízení a také s ovládacím panelem.

S podřízenými moduly deska komunikuje pomocí 3. sériového portu (ttymxc2). Na 5. sériový port je připojen WiFi modul pro komunikaci s ovládacím panelem. Laserový senzor vzdálenosti Sick LMS 100 je jako jediný senzor připojený přímo k řídicí jednotce přes Ethernet. Senzor produkuje velký objem dat [5] a kapacity podřízených modulů nejsou pro jeho obsluhu dostatečné [3].

Konzole operačního systému je přesměrována rovněž na sériový port (ttymxc0) dostupný ve formě CANNON konektoru s rozhraním RS232 a také na konektoru USB přes FT232RQ [6].

2.2.2 Podřízené moduly

Komunikace mezi i.MX31 a podřízenými moduly probíhá přes sběrnici CAN, při čemž jako aplikační vrstva je použita podmnožina protokolu CANopen.. CAN není implementována přímo na desce i.MX31. V tomto případě modul napájení tvoří most mezi sériovým portem (ttymxc2) a sběrnicí CAN.

Jednotlivé podřízené moduly sbírají informace ze senzorů a tyto naměřená data posílají řídící jednotce. Výjimkou je modul pohonu, který řídí pohon vozidla.

Distribuce dat je založena na sdílené paměti (i.MX31), odkud jsou data cyklicky posílána sériovou linkou modulu správy napájení, na které řadič sběrnice CAN data dále distribuuje ostatním podřízeným modulům. Z pohledu řídící desky je komunikace s podřízenými jednotkami velmi jednoduchá, protože stačí zapisovat, respektive číst ze sdílené paměti.

2.3 SOFTWAROVÉ ŘEŠENÍ

Řídicí software běží na operačním systému reálného času Timesys Linux. To umožňuje rozdělit řídící software na jednotlivé procesy. Navíc použitý RTOS poskytuje potřebné drivery pro periferie desky i.MX31 a není nutné je tedy vytvářet samostatně. Použitému RTOS se podrobněji věnuje kapitola 2.5. Na [Obr. 3] jsou znázorněny procesy účastnící se řízení.

Po nabootování operačního systému je spuštěn proces *vehicle_init*. Ten nastartuje všechny ostatní potřebné procesy a po té je ukončen.

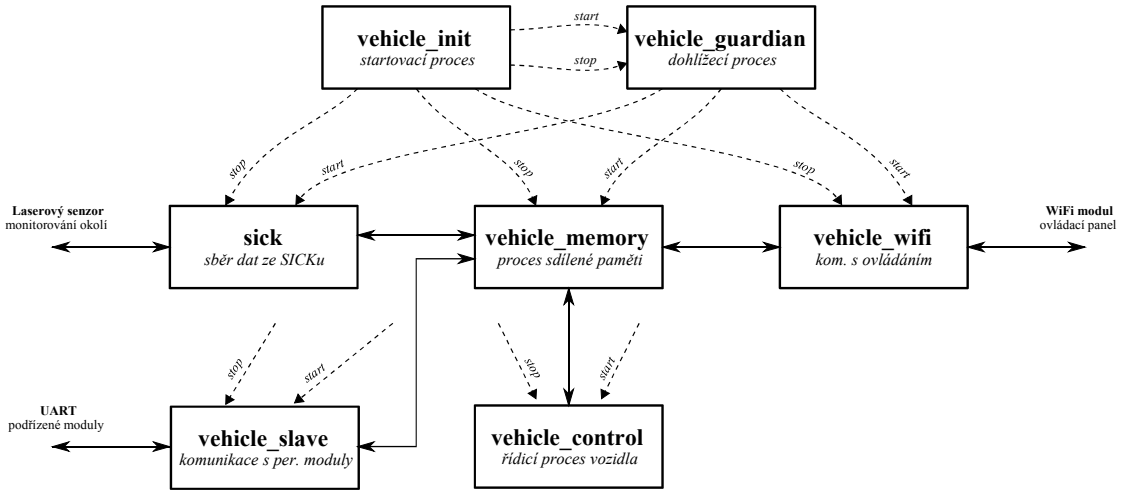
Bezpečnostní proces *vehicle_guardian* sleduje běh ostatních procesů. V případě, že některý proces přestane běžet například odpojením čidla, *vehicle_guardian* znovu jej znovu nastartuje.

Proces *vehicle_memory* zprostředkovává sdílenou paměť ostatním procesům. Paměť se přiděluje procesům na základě globálního mutexu[8].

Získávání a filtrování dat z laserového senzoru obstarává proces *sick*. Získaná data jsou ukládána do sdílené paměti.

Proces *vehicle_slave* zabezpečuje cyklickou distribuci dat ze sdílené paměti do ostatních modulů distribuovaného systému. Obdobně pracuje i proces *vehicle_wifi*. který cyklicky synchronizuje data s identickou pamětí na straně vzdáleného panelu ovládání.

Posledním procesem je *vehicle_control*, který obstarává řízení vozidla. Obsahuje algoritmy vyhodnocování pozice vozidla, zpracování vektorových map, Dijkstrův algoritmus pro plánování trasy a samotné řízení akčních členů tak, aby bylo dosaženo stanovené destinace.



Obr. 3: Schématické znázornění interakce procesů řídicího softwaru [3]

2.4 OVLÁDACÍ PANEL

Pro vizualizaci dat z vozidla, jeho vzdálené ovládání a sledování slouží ovládací panel [Obr. 4]. Je postaven podobně jako řídicí jednotka vozidla na platformě i.MX31 s deskou procesoru SOM-LV[9]. Panel obsahuje barevný LCD displej s rozlišením 640x480 pixelů, vybaveným odporovou dotykovou fólií[10]. Ovládací panel dále obsahuje baterii, WiFi modul s anténou a USB konektor pro připojení joysticku [7],[8].

8 Synchronizační primitivum limitující přístup jednotlivých procesů do společného prostředku, jako třeba paměti.

9 Zkratka pro System on Module, jedná se o desku s mikroprocesorem a několika nutnými periferiemi.

10 Tzv. touchscreen.

Software panelu je postaven na operačním systému Timesys Linux. Pro GUI je využita knihovna Qt Embedded Linux, která pracuje přímo nad framebufferem[11].

Mezi hlavní funkce panelu patří [8]:

- Přehledná vizualizace údajů o vozidle
- Zobrazení údajů ze senzorů
- Vykreslování mapy a vozidla. Pozice vozidla se nastavuje předem a je aktualizována údaji z odometrie.
- Diagnostická obrazovka s grafy (proud do motoru, otáčky motoru, proud odebíraný z baterie, napětí baterie, ping) a možností zobrazení surových dat.

Obr. 4: Ovládací panel průzkumného vozidla

2.4.1 Komunikace ovládacího panelu s vozidlem

O komunikaci se stará již zmíněný proces *vehicle_wifi*. Ten cyklicky posílá, respektive přijímá malé datové rámce podle stanoveného komunikačního protokolu. Komunikační protokol je možné nalézt v příloze. V případě, že je přijata zpráva, je tato zpráva rozparsována a dále je vyhodnocen její typ. Podle typu jsou data zprávy přepsána do sdílené paměti. Zároveň v intervalech 20 ms a 300 ms[12] jsou odesílány zprávy ovládacímu panelu, který přijatou zprávu zpracuje podle stejného schématu a tak je zajištěno zrcadlení těchto pamětí.

11 Obrazové výstupní zařízení; vyrovnávací paměť snímku pro vykreslení na obrazovku.

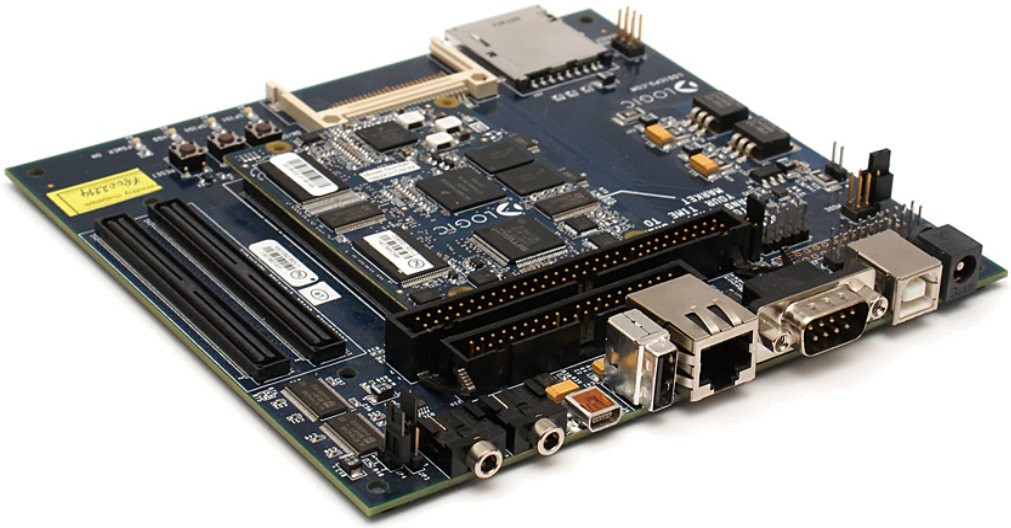
12 Interval odesílání zpráv je dán typem zprávy.

2.5 Řídící jednotka i.MX31 LiteKit a RTOS Timesys Linux

Řízení průzkumného vozidla je založeno na desce i.MX31 LiteKit, na kterém běží operační systém reálného času Timesys Linux. Jako zavaděč[13] je použit LogicLoader[14]. Vzhledem k faktu, že tato práce se zabývá především softwarem, který na této platformě poběží, a tedy seznámení s tímto systémem tvoří značnou část práce, bude v této podkapitole tato jednotka popsána a nastíněn způsob práce s tímto systémem.

2.5.1 Vlastnosti i.MX31 LiteKit

Kit sestává z 3 hlavních desek: deska procesoru SOM-LV[15], deska periferií Application Baseboard a rozšiřující desky, viz [Obr. 5].



Obr. 5: Vývojová deska i.MX31 LiteKit [3]

SOM-LV obsahuje výkonný multimediální procesor fy. Freescale, založený na architektuře ARM a dále všechny bezprostředně potřebné obvody pro funkci procesoru. Deska Application Baseboard obsahuje management napájení, konektory pro připojení ATA disku, grafického displeje, slot pro SD a CF karty a dále standardní porty, jakými jsou USB, LAN, RS232 a zvukový výstup. Dále obsahuje konektor pro připojení poslední, rozšiřující desky. Tato deska poskytuje mimo jiné dodatečné sériové porty a USB porty, dále rozhranní SPI, IIC, dodatečné konektory pro připojení LCD a dotykového panelu.

13 Krátký strojový kód, jehož úkolem je načíst jádro OS.

14 Označován jako LoLo.

15 Zkratka pro System on Module.

Tab. : Základní parametry desky SOM-LV [9]:

Procesor:	Freescale® ARM 1136JF-S i.MX31 taktovaný na 532 MHz
Podporované OS:	Linux BSP Windows CE 5.0 BSP Windows Embedded CE 6.0 BSP
Paměti:	NAND flash 64 MB NOR flash 2 MB Nízko-příkonová 266 MHz DDR SDRAM 128 MB
Připojení:	10/100 Base-T Ethernetový řadič (application/debug) USB 2.0 1x high-speed host interface, 1x On-the-Go device interface 3x UART
Zobrazení:	Vestavěný LCD řadič do rozlišení 800 x 600 x 18ti bitová barevná hloubka Integrovaný řadič 4vodičového touchscreenu
Audio:	IIC kompatibilní audio kodek (Freescale MC13783 – 16bit stereo DAC, 13bit ADC)
Rozšíření paměti:	Podpora ATA, CompactFlash, typ 1 Podpora SD/MMC paměťových karet

1.1 TIMESYS LINUX

Tento operační systém je dodáván s kitem i.MX31 LiteKit. Jedná se o derivát embedded RTOS Linuxu. Pro vývoj aplikačního softwaru je dodáváno rozšíření Timesys IDE pro vývojové prostředí Eclipse [10].

Výhodou použití RTOS je snadné rozdělení aplikace do jednotlivých procesů nebo vláken a podpora driverů ze strany OS. Tyto drivery lze jednoduše používat a to bez podrobné znalosti komunikace s cílovým zařízením. Navíc v případě Linuxu lze hovořit i o zjednodušení přístupu k paměti, periferiím a sběrnicím, jelikož práce s těmito prostředky je stejná jako práce se soubory.

1.2 SPUŠTĚNÍ I.MX31 A JÁDRA OS

Desku i.MX31 je možné z hlediska vývoje aplikace možno propojit s hostitelským počítačem pomocí RS232 nebo Ethernetu. Rámcové spuštění probíhá, respektive je nakonfigurováno následovně:

1. Po zapnutí se spustí zavaděč. V tomto případě je použit LogicLoader.
2. Ten načte skript, ve kterém je uloženo umístění kernelu[16] OS a zapne Ethernet a přes TFTP stáhne jádro do operační paměti.
3. Zavaděč spustí jádro.

16 Jádro OS.

4. Jádro si připojí souborový systém z NFS serveru přes Ethernet z hostitelského počítače.
5. Spuštění samotné uživatelské aplikace.

Tímto způsobem je spuštěna samotná aplikace bez fyzického kopírování souborového systému. Po ukončení vývoje a odladění aplikace se vypálí operační systém do paměti NAND a do paměti NOR se vypálí spouštěcí skript, který obsahuje informaci o umístění souborového systému, který si má jádro připojit. Ten může být umístěn prakticky na libovolném z podporovaných médií, v tomto případě se jedná o SD kartu.

Podrobný popis oživení desky i.MX31, instalaci potřebného softwaru, konfigurace a ladění je uveden v [11].

3 METODY LOKALIZACE S VYUŽITÍM BEZDRÁTOVÝCH SÍTÍ

Bezdrátovou lokalizaci je možné nasadit s větším, či menším úspěchem prakticky ve všech situacích, kdy máme větší počet vysílacích stanic, které můžeme od sebe rozlišit a určit jejich vzdálenost. Rozlišení stanic může probíhat buď na základě různého vysílacího kanálu, tedy nosné frekvence, nebo podle některého jednoznačného identifikátoru (například MAC adresy). Podle způsobu vyhodnocení vzdálenosti od vysílače, popřípadě směru k vysílači, můžeme tyto lokalizační techniky dále dělit na [12]:

- útlumové metody:
  - SNR (Signal to Noise Ratio) – vzdálenost je dána vztahem k poměru výkonu a šumu
  - RSSI (Received Signal Strength Indication) – vzdálenost dána závislostí na síle signálu, respektive na jeho útlumu
- geometrické:
  - AoA (Angle of Arrival) – měří se úhel k vysílači
  - časové:
  - PoA (Phase of Arrival) – měření fázového zpoždění příchozího signálu
  - ToA (Time of Arrival), TDoA (Difference of Arrival) – vzdálenost je dána časovým zpožděním mezi vysláním a příjmem

Problematika lokalizace s využitím bezdrátových sítí je z části převzata z bakalářské práce [2], podrobnější informace je možné nalést v ní.

3.1 ROZBOR TECHNOLOGIÍ POUŽITELNÝCH K LOKALIZACI

GSM

GSM sítě jsou velmi dobře rozšířené a jejich pokrytí je v ČR téměř po celém území. Pro lokalizaci se obvykle využívá metoda TDoA. V případě, že lokalizace je prováděna ze strany GSM sítě, tedy věží, používá se metoda AoA. Běžně je uživatelům dostupná lokalizace TDoA, kterou dokáže určit lokalizovaný přístroj sám. Někteří operátoři umožňují i lokalizaci cizích přístrojů[17]. Tato pak probíhá na základě AoA.

Přesnost GSM lokalizace je velmi nízká. V místech s velkou hustotou sítě (typicky velká města) je přesnost v řádech desítek až stovek metrů, v oblastech s malou hustotou sítě (volné krajiny) může přesnost lokalizace klesnout až na jednotky kilometrů.

17 Například T-mobile poskytuje službu „T-mobile Locator“, nebo nově službu „Kde je…“

Za použití metody fingerprints[18], lze v hustě pokrytých oblastech dosáhnout přesnosti i v řádech metrů [13].

GPS a Galileo

Lokalizace je založena na metodě ToA, tedy z doby letu signálu se vyvozuje vzdálenost lokalizovaného zařízení od satelitu. Přesnost systému je až v řádu desítek cm, ale pro civilní uživatele je přesnost omezena na asi 10 m. GPS pracuje na frekvencích v pásmu 1176,45 MHz (L5) až 1841,40 MHz (L4), tedy v pásmu UHF a tedy prakticky na přímou viditelnost. V budovách nebo silně zalesněných oblastech obvykle není dostatečný signál a lokalizace není možná.

Pro určení pozice se využívá multilaterace. Pro určení pozice na zemském povrchu (zeměpisná šířka a délka) je potřeba signálu alespoň ze tří satelitů. Pokud jsou k dispozici signály z minimálně 4 satelitů, je možné určit i nadmořskou výšku.

Bluetooth a ZigBee

Bluetooth pracuje v pásmu 2,4 GHz, ZigBee může pracovat v pásmech kolem 1 GHz a na 2.4 GHz. Dosah signálu je mezi 10 až 100 m v případě Bluetooth a 10 až 50 m v případě ZigBee. Obě technologie kladou důraz na spotřebu a tedy zařízení nejvyšší výkonové kategorie (a tedy i dosahu) jsou instalovány jen zřídka.

Z důvodů nízkého dosahu je využití lokalizace obvykle omezeno na jednotlivé oblasti s vysílačem. Tedy pouze informace u kterého vysílače se zařízení nachází. Přesnější lokalizace vyžaduje instalaci výkonnějších vysílačů.

RFID čipy

V poslední době se začíná hovořit o použití RFID pro účely lokalizace. RFID jsou malá zařízení s anténou a mikrokontrolérem bez zdroje energie. Při potřebě vyčíst informace z RFID čipu je nejprve vysílán signál, kterým se RFID nabije a následně z této energie pošle přijímač zpět data vysílači. Lokalizace je omezena pouze na informaci u které čtečky se RFID nachází.

WLAN sítě

WLAN, tedy Wireless LAN sítě jsou sítě standardů IEEE 802.11. Dosah bývá obvykle kolem 100 m, pro velké směrové antény lze získat dosah i v kilometrech. Každá AP WiFi sítě obsahuje svou unikátní MAC adresu a při dostatečném pokrytí lze dosáhnout přesnosti až 1 metr. Pro určení vzdálenosti lokalizovaného zařízení od AP se využívá SNR nebo častěji RSSI, které WiFi nativně poskytují.

3.1.1 Zhodnocení a výběr vhodného řešení

Lokalizace v případě GSM není dostatečně přesná, pro její zpřesnění by sice mohly být použity fingerprinty, ale pro tuto aplikaci to není vhodné[19].

18 Lokalizační metoda založená na mapování skutečné hodnoty RSSI. RSSI hodnoty v daném místě se následně porovnávají s hodnotami získanými mapováním

19 Pro získání fingerprintů je potřeba nejprve v dané oblasti provést důkladná měření.

GPS je ve vozidle již použito, ale není použitelné v budovách. Systém Galileo naráží na stejný problém, a tedy opět toto řešení není použitelné.

Lokalizace pomocí RFID vyžaduje instalaci velkého množství čteček a je vhodné například pro monitorování zboží, nicméně není určeno pro „spojitou“ lokalizaci.

Technologie Bluetooth a ZigBee dosahují pro lokalizaci podobných vlastností jako WiFi a jsou docela běžně používány v průmyslu. Nicméně obecně rozšíření dostatečné není a muselo by být v místě plánovaného nasazení vozidla předem nainstalováno. To není přípustné a bude pro to zvoleno WiFi, kde je možné využít již dnes téměř všudypřítomnou infrastrukturu.

3.2 MODELOVÁNÍ ŠÍŘENÍ WIFI SIGNÁLU

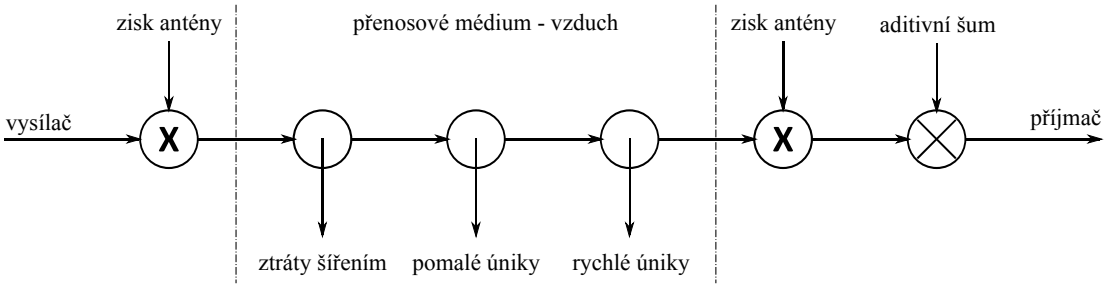
WiFi sítě jsou definovány standardy IEEE 802.11 a pracují v bezlicenčních pásmech ISM. Dnes nejčastěji používaná nosná frekvence je 2,4 GHz[20]. Legislativa České republiky definuje v tomto pásmu celkem 13 kanálů o šířce pásma 22 MHz a rozestupem mezi středy kanálů 5 MHz. WiFi signály lze z frekvenčního hlediska kategorizovat jako rozptýlené prostorové přízemní elektromagnetické vlny [18].

Signál se může mezi vysílačem a přijímačem šířit na přímou viditelnost nebo s odrazy, ohyby a rozptylem na překážkách. Použité pásmo UHF a uvažování relativně krátkých spojů pro WiFi také prakticky eliminuje celou řadu dalších jevů, jako jsou útlumy hydrometeory, troposférická refrakce, útlum atmosférickými plyny, aj. Pro modelování můžeme použít popis pomocí intenzity elektromagnetického pole, nebo jako výkonovou úroveň definovanou v dB.

Z hlediska umístění antén a z toho vyplývajících mechanismů tlumení signálu můžeme rádiový spoj dále dělit na pevný a mobilní. Pro tuto práci bude dále uvažován mobilní spoj.

3.2.1 Modelování výkonové bilance mobilního spoje

Signál mobilního spoje je svým šířením tlumen a působí na něj několik fyzikálních mechanismů. Tuto výkonovou bilanci demonstruje [Obr. 6].



Obr. 6: Mechanizmus modelování výkonové bilance rádiového přenosu v zástavbě

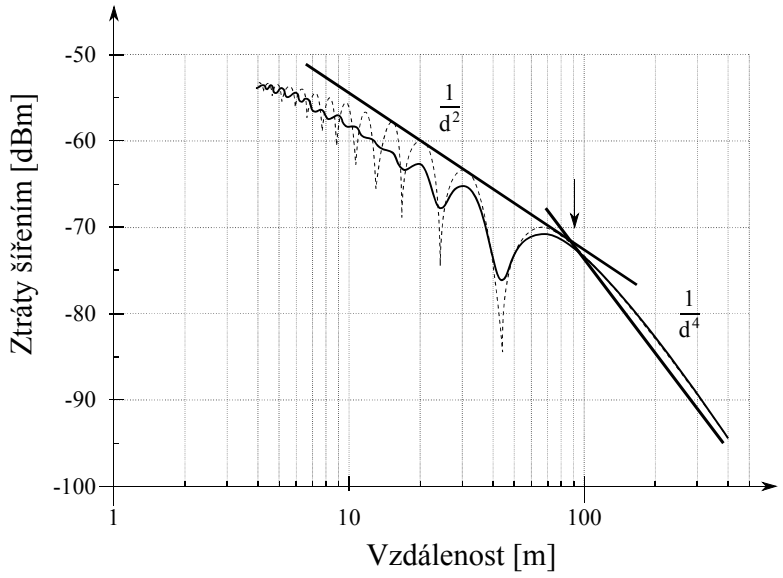
20 IEEE 802.11a a IEEE 802.11n pracují na pásmu 5 GHz

Z výkonové bilance je patrné, že se na výkonu signálu silně uplatňuje zisk přijící i vysílací antény. Všechny šumy způsobené interferencí a samotným šumem přijímače i vysílače se uvažují jako aditivní na straně přijímače. Vlastní útlum šířením signálu je složen ze tří složek [14]:

- ztráty šířením – útlum závislý na délce spoje a typu prostředí. Je časově invariantní a z pohledu lokalizace je to složka nesoucí informaci.
- pomalé úniky – útlum způsobený zastíněním spoje překážkami. Z hlediska vlnové délky dochází k relativně pomalému kolísání a pokles může být velmi velký.
- rychlé úniky - způsobují velmi rychlé a hluboké kolísání úrovně signálu. Dochází k nim především vícecestným šířením signálu a Dopplerovým posuvem[21], který vzniká v důsledku pohybu mobilní antény a okolních objektů. Tyto úniky jsou časově variantní a kmitočtově selektivní. Vykazují Rayleighovo rozložení[22] v čase.

3.2.2 Ztráty šířením

Signál šířící se v poloprostoru vykazuje do jistého bodu označovaného jako Frencisův zlom závislost nepřímo úměrnou kvadrátu vzdálenosti (útlum 20 dB/dek). Po tomto bodě signál vykazuje ztrátu nepřímo úměrnou 4. mocnině (40 dB/dek). Situace je zobrazena na [Obr. 7]. K Frencisovu zlomu dochází v místě, kde je dráhový rozdíl vysílaného a odraženého signál roven polovině vlnové délky [14].



Obr. 7: Průběh spádové křivky s vyznačeným Frencisovým zlomem

21 Změna vlnové délky vysílaného a přijímaného signálu způsobená nenulovou vzájemnou rychlostí.

22 Spojité nesymetrické rozložení pravděpodobnosti podobné Gaussovu.

Při malých výškách antén v porovnání s jejich vzdáleností, můžeme vypočíst Francisův zlom podle vztahu [14]:

$$d_f = \frac{4\, h_1\, h_2}{\lambda} \tag{1}$$

kde: d_f Francisův zlom (m)
h_1 a h_2 výsky antén spoje (m)
λ vlnová délka (m)

Pro konkrétní nasazení lokalizace můžeme Francisův zlom vyčíslit (uvažujeme výšku antény vozidla 0,7 m, výška AP 2,5 m):

$$\lambda = \frac{3 \cdot 10^5}{f} = \frac{3 \cdot 10^5}{2{,}4 \cdot 10^6} = 0{,}125\ m \tag{2}$$

$$d_f = \frac{4\, h_1\, h_2}{\lambda} = \frac{4\, h_1\, h_2}{0{,}125} = 56\ m \tag{3}$$

Vzhledem k uvažovanému dosahu lze konstatovat, že lokalizace se bude pohybovat vždy před Francisovým zlomem. Tento fakt bude následně v algoritmu lokalizace podpořen i vyřazením AP s příliš nízkým signálem (bude rozebráno kapitole Chyba: zdroj odkazu nenalezen).

Základní empirický model

Pro prostředí bez překážek, kde předpokládáme hladkou spádovou křivku, můžeme použít k predikci ztrát šíření základní empirický model [15]:

$$L(d) = L_1(d_1) + 10\, n \log \frac{d}{d_1} \qquad d \geq d_1 \tag{4}$$

kde: L(d) ztráty šířením (dB)
n spádový koeficient (-)
d vzdálenost vysílací a přijicí antény (m)
d_1 referenční vzdálenost (m)
L_1 referenční útlum při d_1 (dB)

Úpravou základního empirického modelu lze dostat Path-Lost model, který dále vztah rozepisuje o zisk antén, ztráty šířením a pomalé úniky:

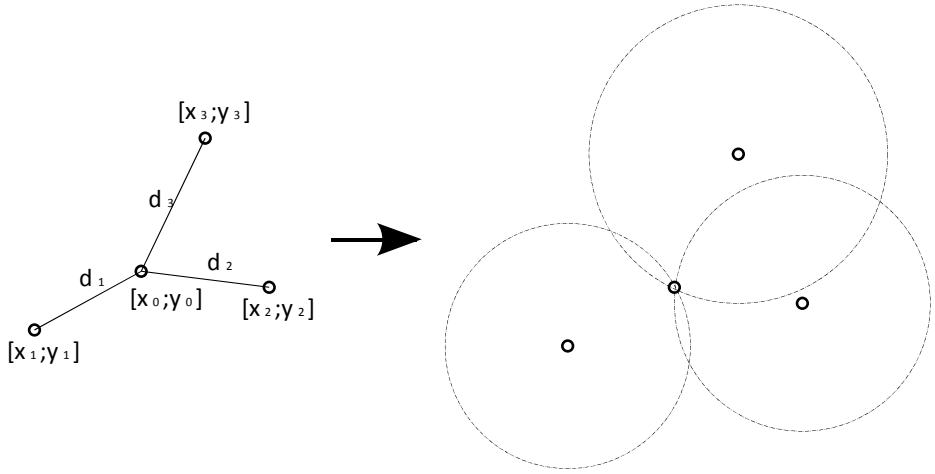
$$r = t - l_0 + 10\, n \log \frac{d}{d_1} - S \qquad d \geq d_1 \tag{5}$$

kde:
- r výkon na straně přijímače, včetně zisku antény přijímače (dBm)
- t efektivní vyzářený výkon, včetně zisku antény vysílače (dBm)
- l_0 ztráty šířením (dBm)
- n spádový koeficient (-)
- d vzdálenost vysílací a přijící antény (m)
- d_1 referenční vzdálenost (m)
- S ztráty pomalými úniky, log-normal shaddow fading (dB)

3.3 LOKALIZAČNÍ METODY

V této části práce bude uvedena některé lokalizační metody založené na vyhodnocování vzdáleností, tzv. “Range Based Network Localization”.

3.3.1 Multilaterace



Obr. 8: Princip multilaterace

Velká většina lokalizačních metod využívá tzv. multilaterace. Jedná se o matematický aparát, který na základě vzdáleností od známých referenčních bodů stanoví pozici bodu hledaného[23] [Obr. 8].

23 Dále bude namísto slova bod používáno slovo uzel, jak tomu bývá v literatuře

Výpočet multilaterace můžeme popsat analyticky soustavou rovnic (analytickým zápisem kružnic):

$$\begin{aligned} d_1^2 &= (x_1 - x_0)^2 + (y_1 - y_0)^2 \\ d_2^2 &= (x_2 - x_0)^2 + (y_2 - y_0)^2 \\ &\vdots \\ d_n^2 &= (x_n - x_0)^2 + (y_n - y_0)^2 \end{aligned} \tag{6}$$

Známe-li pozici alespoň 3 uzlů a jejich vzdálenosti k neznámému uzlu, můžeme vypočítat pozici neznámého uzlu $[x_0;y_0]$[24]. Přeuspořádáním soustavy lze získat maticový zápis:

$$\mathbf{H}\,\bar{\mathrm{x}} = \bar{\mathrm{b}}$$

$$\text{kde} \quad \mathbf{H} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_n - x_1 & y_n - y_1 \end{bmatrix} \quad \bar{\mathrm{x}} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad \bar{\mathrm{b}} = \frac{1}{2} \begin{bmatrix} x_2^2 + y_2^2 - d_2^2 - (x_1^2 + y_1^2 - d_1^2) \\ x_3^2 + y_3^2 - d_3^2 - (x_1^2 + y_1^2 - d_1^2) \\ \vdots \\ x_n^2 + y_n^2 - d_n^2 - (x_1^2 + y_1^2 - d_1^2) \end{bmatrix} \tag{7}$$

Přenásobením soustavy inverzní maticí **H** zleva získáme vztah, ze kterého můžeme přímo vypočíst souřadnice neznámého uzlu x [16]:

$$\bar{\mathrm{x}} = \mathbf{H}^{-1}\,\bar{\mathrm{b}} \tag{8}$$

Nicméně v praxi je k vzdálenostem vázána chyba. Tato chyba způsobuje, že soustava rovnic je nekompatibilní[25] a $\bar{\mathrm{x}} \notin \Re(\mathbf{H})$. Kružnice se nemusí vůbec protnout, nebo se protínají ve více bodech a proto tento jednoduchý výpočet nelze použít. Rozdíl mezi naměřenou vzdáleností a vzdáleností skutečnou lze popsat vztahem:

$$e_i = d'_i - d_i \qquad \text{kde } i = 1, 2, \ldots, n \tag{9}$$

Tento problém je znám jako „šum vzdálenosti“ a lze ho řešit aplikací některé z aproximačních metod. Pro tuto práci byla vybrána metoda nejmenších čtverců, která je jednoduchá na implementaci a má výhodné vlastnosti. Metoda nejmenších čtverců hledá opravu reziduem $\bar{e}$ vektoru pravé strany $\bar{b}$, minimalizující:

24 Za předpokladu, že uzly jsou lineárně nezávislé a dimenze hledaného řešení je 2, viz. Frobeninova věta.

25 Rovnice se navzájem vylučují.

$$\bar{e} = \min \|\mathbf{H}\,\bar{x} - \bar{b}\| \qquad e \in \mathbb{R}^n \tag{10}$$

Tato podmínka bude splněna, pokud bude mít vektor $\bar{e}$ minimální normu[26] a tedy bude ortogonální[27] na obor hodnot $\Re(\boldsymbol{H})$. Musí platit podmínka [17]:

$$\bar{e} \perp \Re(\mathbf{H}) \Leftrightarrow \mathbf{H}^{\mathbf{T}}\,\bar{e} = 0 \tag{11}$$

Dále můžeme reziduum dosadit:

$$\begin{aligned} &\mathbf{H}^{\mathbf{T}}(\mathbf{H}\,\bar{x} - \bar{b}) = \mathbf{H}^{\mathbf{T}}\,\mathbf{H}\,\bar{x} - \mathbf{H}^{\mathbf{T}}\,\bar{b} = 0 \\ &\mathbf{H}^{\mathbf{T}}\,\mathbf{H}\,\bar{x} = \mathbf{H}^{\mathbf{T}}\,\bar{b} \end{aligned} \tag{12}$$

Obecná formule pro výpočet multilaterace je

$$\bar{b} = \left(\mathbf{H}^{\mathbf{T}}\,\mathbf{H}\right)^{-1}\mathbf{H}^{\mathbf{T}}\,\bar{x} \tag{13}$$

Aby byl přechod z rovnice (12) na (13) platný, musí mít součin $\mathbf{H}^T\mathbf{H}$ inverzi. Díky vlastnostem tohoto součinu bude výsledná matice vždy čtvercová, a tedy vztah bude platný pro prakticky libovolný počet uzlů při platnosti výše uvedené podmínky. Z definice matice **H** na (7) je patrné, že součin $\mathbf{H}^T\mathbf{H}$ bude regulární a bude mít inverzi právě tehdy, pokud bude matice **H** mít hodnost větší, než je dimenze hledaného řešení. Tedy pro jednoznačné určení pozice neznámého uzlu potřebujeme alespoň 3 lineárně nezávislé referenční uzly (neležící na přímce) pro určení pozice na ploše (2D) a alespoň 4 nezávislé referenční uzly pro určení pozice v prostoru (3D).

3.3.2 Gaussův filtr

Měřená hodnota RSSI, tedy útlumu signálu z jednotlivých AP je velmi nestálá a v čase variantní. Těmto změnám se říká rychlé úniky signálu a pocházejí z různých zdrojů, viz kapitola 3.2.1. Chyba způsobená rychlými úniky může být velmi velká, řádově v jednotkách až desítkách dBm. Aby bylo možné RSSI vůbec vyhodnocovat, je nutné tyto úniky filtrovat. Jednou z metod vhodných k filtraci těchto signálů je Gaussův filtr [18].

Gaussův filtr má tyto základní vlastnosti:

- Odezva na Diracův impuls (impulsní charakteristika) je Gaussova funkce
- Nevytváří překmity při odezvě na Heavysideův skok
- Má minimální možné skupinové zpoždění
- Fourierova transformace Gaussovy funkce je opět Gaussova funkce

26 Euklidovskou vzdálenost

27 pravoúhlý na vektor oboru hodnot

Gaussův filtr je z praktického hlediska typ digitálního filtru, kde koeficienty jsou dány hodnotami Fourierovy transformace Gaussovy funkce. Díky vlastnosti Gaussovy funkce tvoří koeficienty filtru přímo hodnoty této funkce. Výstupní signál se počítá jako konvoluce vstupního signálu s Gaussovou funkcí. Jednorozměrná Gaussova funkce má předpis:

$$f_G(x) \stackrel{\text{def}}{=} \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{14}$$

kde: σ směrodatná odchylka
μ střední hodnota (maximum Gaussovy funkce)

Celý Gaussův filtr lze pak popsat konvolucí:

$$(f * g)(x) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\alpha) \cdot g(x-a)\ \mathrm{d}\alpha \tag{15}$$

Nebo v diskrétní oblasti jako

$$(f * g)(x) \stackrel{\text{def}}{=} \sum_{i=0}^{k} f(i) \cdot g(k-i) = \sum_{i=0}^{k} g(i) \cdot f(k-i) \tag{16}$$

Pro praktické použití Gaussova filtru se používá jen omezený počet vzorků Gaussovy funkce. Takto navzorkovaná Gaussova funkce pak tvoří vektor, který se nazývá kernel. Při aplikaci filtru se pak uchovává historie hodnot filtrovaného signálu o hloubce dané počtem prvků kernelu. Při každém vzorku se historie aktualizuje, vynásobí kernelem a získané součiny sečtou. Je důležité zvolit správnou hodnotu σ, která ovlivňuje kvalitu filtrace, ale zároveň vnáší zpoždění a setrvačnost. Při $\sigma \to \infty$ Gaussův filtr přechází na klouzavý průměr.

3.4 PRAVDĚPODOBNOSTNÍ METODY LOKALIZACE

Pravděpodobnostní metody lokalizace patří mezi velmi populární a typicky i rychlé. Jejich předností je, že dokáží pracovat s pozicí včetně její nejistoty.

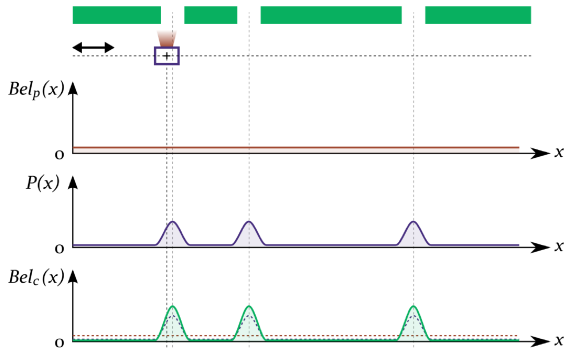
V této kapitole bude nejdříve rozebrán princip metody známé jako Kalmanův filtr a následně lokalizace Monte-Carlo[28].

28 Dále pod zkratkou MCL.

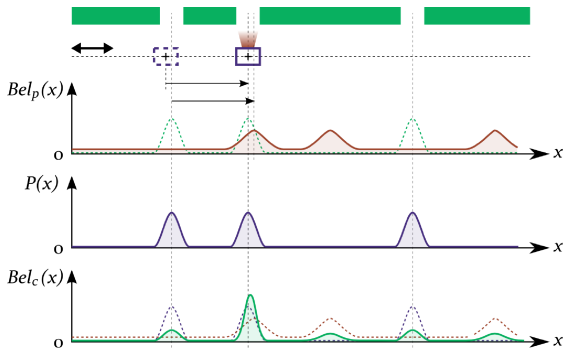
3.4.1 Kalmanův filtr

Kalmanův filtr patří do rodiny pravděpodobnostních metod Markovské lokalizace. Jadná se o velmi populární lokalizační metody, které reprezentují odhad pozice pomocí unimodálního gaussovského rozložení. Základní princip byl představen již v roce 1960 Rudolphem Kalmanem [19]. Pozice výskytu robota je pak popsána rozptylem a střední hodnotou této distribuce. Použité gaussovské rozložení navíc umožňuje pracovat s příslušnou neurčitostí, nejistotou pozice robotu v prostoru. Tato reprezentace pravděpodobnosti je nazývána důvěrou, anglicky „belief“ *Bel*(x) [20].

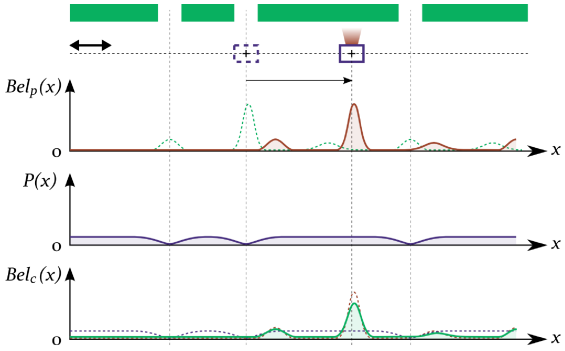
Proces lokalizace je patrný z příkladu na [Obr. 9], [Obr. 10] a [Obr. 11], kde je ukázán příklad jednorozměrné lokalizace s robotem, který se pohybuje po chodbě s dveřmi.



Obr. 9: Robot nezná pozici - apriorní pravděpodobnost je konstantní. Změří senzorem okolí a detekuje dveře. Pravděpodobnost výskytu před dveřmi je tedy maximální.



Obr. 10: Robot provede akci – přemístí se a tuto akci ve formě modelu pohybu aplikuje i na svou „důvěru“ v pozici Bel(x). Následně robot opět aplikuje data ze senzorů. Bel(x) obsahuje výrazné globální maximum a robot je lokalizován.



Obr. 11: Situace po novém pohybu robota

Proces lokalizace se je tvořen aktualizacemi gaussovského rozložení následovně [20]:

- Robot se přemístí a podle předpokládané velikosti a směru pohybu je posunuto i gaussovské rozložení, které je typicky měřeno odometrickými senzory, nebo je určena na základě akčních zásahů do motorů robotu (model pohybu). Současně s translací pozice je upraven i rozptyl podle modelu odometrického subsystému, popřípadě podle jeho předpokládaných chyb. Tato fáze se obvykle nazývá predikce.
- Následně jsou zahrnuty senzorické informace ze senzorů, které nesou údaje o skutečné pozici ve vztahu robota k modelu skutečného světa (model senzoru) a aplikovány ve formě jisté korekce do gaussovského rozložení. Využití informací ze senzorů zvýšenou neurčitost (rozptyl) gaussovského rozložení danou predikcí opět sníží a cyklus se opakuje.

Formálně lze činnost Kalmanova filtru, popsat následujícími vztahy [21]:

$$\mu'_{t-1} = \mu_{t-1} + \mathbf{B}\,\mu_t \tag{17}$$

$$\Sigma'_{t-1} = \Sigma_{t-1} + \Sigma_{control} \tag{18}$$

$$K_t = \Sigma'_{t-1}\,\mathbf{C}^{\mathbf{T}}\left(\mathbf{C}\,\Sigma'_{t-1}\,\mathbf{C}^{\mathbf{T}} + \Sigma_{measure}\right)^{-1} \tag{19}$$

$$\mu_t = \mu'_{t-1} + K_t\left(z - \mathbf{C}\,\mu'_{t-1}\right) \tag{20}$$

$$\Sigma_t = \left(1 - K_t\,\mathbf{C}\right)\Sigma'_{t-1} \tag{21}$$

kde:

μ_t a Σ_t	momenty distribuce pravděpodobnosti
B	převodní matice zadání
C	převodní matice měření
$\Sigma_{control}$	předpokládaná přesnost pohybového modelu
$\Sigma_{measure}$	„důvěra“ v senzorická data
K_t	Kalmanovo zesílení
z	vektor naměřených dat

Kalmanovo zesílení K_t udává, jakou měrou bude budoucí odhad polohy robotu (stav) ovlivněn novými senzorickými daty a jeho velikost závisí na odhadované přesnosti minulého odhadu stavu Σ_{t-1} a přesnosti měření $\Sigma_{measure}$.

Pro úplnost je vhodné zmínit, že výše uvedené vztahy jsou platné pouze pro lineární systémy, pro praktické realizace je nutné využít rozšířený Kalmanův filtr, který je charakteristický tím, že řešený problém linearizuje vzhledem k aktuální pozici robotu.

Výhoda metod, které jsou založeny na Kalmanově filtru je především v jejich vysoké přesnosti (za předpokladu, známé výchozí pozici) a v efektivitě. Nevýhody lze nalézt pak především v omezení, které jsou kladeny reprezentací distribuce hodnot pravděpodobnosti. Teoreticky lze modelovat vzhledem k použité distribuci všechny procesy s normálním rozložením. Nicméně toto je v praxi splněno výjimečně a zároveň z tohoto důvodu není možné řešit úlohu globální lokalizace a metoda není schopna se zotavit z větších pozičních chyb, či fatálních chyb, jako je „problém nakopnutého robotu“[29] [21].

Uvedené nevýhody pramenící z použité distribuce řeší dále představená metoda Monte Carlo.

1.2.1 Metoda Monte Carlo

Metoda Monte-Carlo reprezentuje distribuci pravděpodobnosti polohy množinou částic - vzorků. Tento typ reprezentace distribuce je poměrně často používaný v mnoha různých oborech. Je známá také jako *částicový filtr* (*particle filter*), *kondenzační algoritmus* (zejména v oblasti počítačového vidění) a také jako *bootstrap filters*, *interacting particle approximations* nebo *survival of the fittest* [20].

Výhody použití filtrů z rodiny Monte-Carlo lze spatřit v následujících bodech [21]:

- Částicové filtry jsou univerzálním nástrojem pro aproximaci hustot pravděpodobností a vzhledem k použité reprezentaci distribuce nekladou omezení na tvar posteriorních hustot pravděpodobnosti.
- Jsou velmi flexibilní z hlediska přizpůsobení charakteristikám senzorů, rozložení šumu, či dynamice pohybu.
- Soustředí výpočetní výkon zejména do okolí vysokých pravděpodobností, protože uvnitř stavového prostoru soustředí do těchto míst částice v poměrech odpovídající příslušným posteriorním pravděpodobnostem. Toto vlastnost je podpořena fází převzorkování, viz. níže.
- Výpočetní náročnost algoritmu může být přizpůsobena aktuální výpočetní kapacitě i za běhu přizpůsobením počtu vzorků.
- Implementace může být poměrně snadno paralelizována.

29 Přemístění robota, aniž by to zaregistroval svými senzory.

Základní myšlenkou MCL je vhodná aproximace důvěry $Bel(x)$ váženou množinou vzorků tak, aby diskrétní distribuce definovaná vzorky skutečně odpovídala výchozí spojité distribuci.

Počáteční rozložení důvěry je reprezentováno uniformním rozdělením množiny vzorků m, kterých je M s váhou M^{-1}.

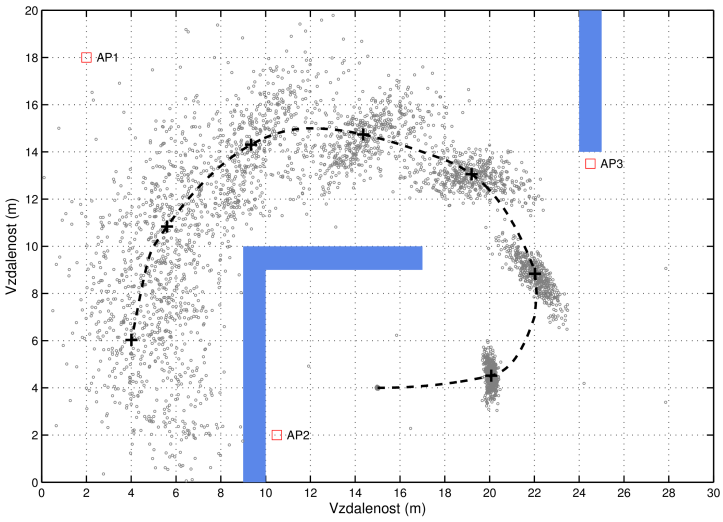
MCL algoritmus v principu aktualizuje stav vytvořením nové množiny vzorků ze stávající množiny jako odezvu na příchozí informaci o pohybu robotu u_{t-1} (predikci) a ze senzorických dat z (korekci) následovně [21]:

1. Vybere se náhodně vzorek x_{t-1} z aktuálního popisu důvěru $Bel_{t-1}(x_{t-1})$.
2. Pro vybraný vzorek x_{t-1} je odhadnuta jeho nová možná pozice x_t podle rozložení pravděpodobnosti $P(z_t \mid x_t, m)$ z modelu pohybu (odometrie, akční zásahy).
3. Vzorku x_t je přiřazena výchozí hodnota váhy podle rozložení modelu senzoru a je přidána do množiny vzorků reprezentující důveru $Bel_t(x_t)$.

Kroky 1-3 iterují přes všechny vzorky stavového prostoru množiny $Bel_t(x_t)$ a nakonec jsou normalizovány jejich váhy tak, aby jejich součet byl roven jedné. Shrneme-li výše uvedená fakta, je možné popsat činnost MCL do následujících kroků:

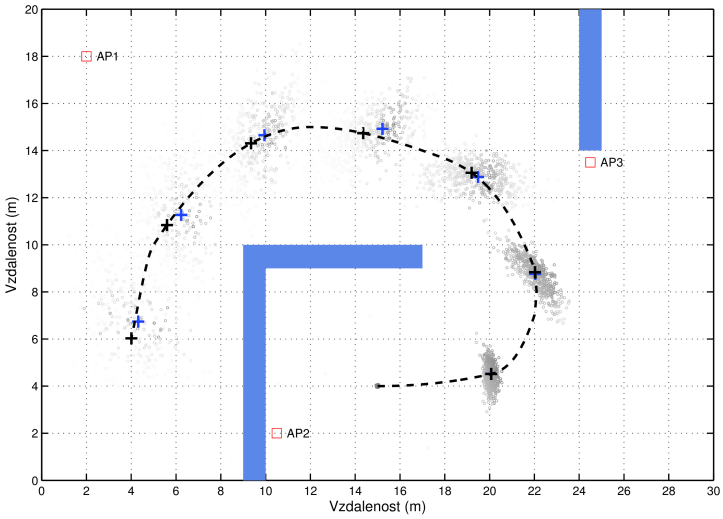
- **Predikce** – translace všech vzorků na základě informací o změně pozice robota např. z odometrie.
- **Korekce** – úprava množiny jednotlivých vzorků a jejich vah na základě shody či neshody naměřených dat s očekáváními, která by odpovídala pozici reprezentované příslušným vzorkem.

Jednotlivé kroky algoritmu ilustrují obrázky [Obr. 12] a [Obr. 13]. První obrázek ilustruje situaci, kdy počáteční poloha je známa a pozice je získávána pouze z modelu akcí (pohybu). Následuje pohyb vozidle po naplánované trajektorii, měřené odomerií. Díky akumulaci chyb je po několika krocích oblast pravděpodobného výskytu již natolik rozsáhlá, že získaná poloha je prakticky nepoužitelná.



Obr. 12: MCL pracující pouze s odometrií (predikcí)

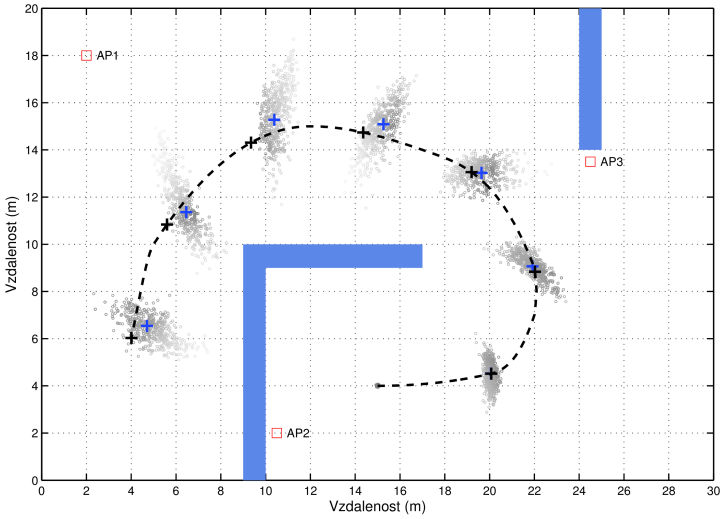
Akumulaci chyb tak jak je to předvedeno na [Obr. 12] je možné omezit zavedením korekce senzorickým modelem. Vzorky jsou tímto modelem váženy a vzorky s největší vahou jsou koncentrovány do míst pravděpodobného výskytu. Situaci demostruje [Obr. 13].



Obr. 13: Aplikace korekce vážením vzorků senzorickým modelem

Protože akumulací chyb dochází k rozptylu vzorků a jejich následnou korekcí k jejich vážení, dochází k rozložení výkonu do velké oblasti. Velká většina má zároveň minimální váhu a výslednou pozici ovlivňují pouze minimálně. Toto chování není žádoucí. Zavedením převzorkování za korekci dosáhneme zvýšení koncentrace vzorků do míst pravděpodobného výskytu (tedy do míst, které nás nejvíce zajímají). Převzorkování probíhá vygenerováním nové množiny vzorků s vahou n^{-1} tak, že

výsledné rozložení pravděpodobností je shodné se stavem před převzorkováním. Situaci opět demonstruje obrázek:



Obr. 14: Kompletní MCL po aplikaci převzorkování

V kontextu praktického použití MCL algoritmu je třeba poznamenat, že díky vzorkovaným aproximacím pravděpodobnostních rozložení algoritmus nemůže pracovat s idealizovanými daty (distribuce $P(z_t \mid x_t, m)$ mají velmi ostrá maxima) a bezchybným modelem světa. Data musí obsahovat určité množství šumu, které více rozprostře distribuce $P(z_t \mid x_t, m)$. Z tohoto důvodu je přímé využití metody pro přesné senzory s nízkým šumem omezeno.

4 KONCEPT ŘEŠENÍ

Vozidlo samotné je koncipováno jako prototyp, platforma poskytující prostor pro další inovaci a rozšíření. Po aplikační stránce činnost vozidla řídí sada víceméně nezávislých programů, které si vyměňují data prostřednictvím sdílené paměti a nad jejich chodem dohlíží proces *vehicle_guardian*, viz. kapitola [2.3].

Aby bylo dosaženo jisté kompaktnosti a kontinuity vyvinutého modulu lokalizace s ostatními systémy vozidla, jsou na řešení kladeny následující požadavky:

- Minimální propojení s ostatními programy vozidla – veškerá komunikace bude probíhat přes sdílenou paměť.
- Kompaktní a samostatný modul s minimálními nároky na zásah do existujících procesů vozidla a kontrolovaný procesem *vehicle_control*.
- Modulární – v případě potřeby musí být snadné integrovat nová senzorická data, jiný formát map, další vlastnosti modelované v rámci MCL, apod. přidáním/nahrazením dané třídy. To implikuje požadavek na oddělení jednotlivých funkčních celků předem danými rozhraními.
- Aplikace by měla splňovat požadavky na soft real-time – výsledek dodaný po deadline je akceptovatelný, ale jeho použití degraduje s rostoucím zpožděním.

Aby bylo dosaženo výše uvedených požadavků, byl koncept lokalizačního modulu stanoven následovně:

Podobně jako ostatní procesy ve vozidle je modul lokalizace napsán v C++ a zkompilovaný do jediného spustitelného souboru[30], který je spouštěn procesem *vehicle_control*.

Základem systému lokalizačního modulu je jádro MCL, které se stará o samotnou lokalizaci a obsahuje algoritmy pro predikci, korekci a převzorkování nad bankou vzorků. Vypočtená výsledná pozice je uložena do sdílené paměti.

Data pro jednotlivé fáze výpočtu jsou získávána z patřičných modelů, které generují komponenty zpracování senzorických dat. Aby bylo dosaženo co největší modularity je použita technika registrování komponent. Jednotlivé komponenty jsou připojeny k jádru MCL předáním instance registrační metodě, která si uloží jejich odkaz do kontejneru „chytrých“ ukazatelů[31]. Jádro MCL je pak skrze tento odkaz volá a přidání nové komponenty pak spočívá pouze v jejích zaregistrování.

Senzorická data, jako jsou údaje z analogových a ultrazvukových dálkoměrů, laserového senzoru SICK LMS 100 jsou vyčítána ze sdílené paměti. Pro každý typ senzorických dat existuje třída,

30 To nevylučuje případné závislosti na knihovnách.

31 Tzv. smart pointer.

která je zpracovává, generuje senzorický model a k vlastnímu jádru MCL je připojena přes rozhraní[32]. Stejně tak jsou řešeny i zdroje pro model(y) pohybu, zde se jedná zejména o odometrii pro prediktivní fázi lokalizace.

Mimo jádra MCL a komponent zpracování senzorických dat obsahuje modul lokalizace i subsystém map s jehož pomocí jsou filtrovány částice, které nemají platnou pozici (např. leží ve zdi).

Přestože proces lokalizace není kritický, je vhodné Požadavek na soft real-time je logický, protože výstupem je pozice vozidla, která má největší užitnou hodnotu v čase, kdy je na ní dotazováno. Latence dodání výsledku způsobuje nárůst nejistoty pozice, tzn. pozice již nemusí být aktuální (typicky vozidlo urazilo během výpočtu určitou vzdálenost). Zároveň je neopodstatněné klást požadavky přísnější s ohledem na prostředí – samotný operační systém je soft real-time[33] a aktuální procesy ve vozidle taktéž nejsou real-time. Pro dosažení požadavku na soft real-time je proces lokalizace škálovatelný z hlediska využití prostředků platformy. Toho je docíleno řízením množství generovaných částic v algoritmu převzorkování MCL. Snížením se úměrně sníží i využití procesorového času a paměti[34] za cenu možného zhoršení přesnosti lokalizace.

Pro efektivní předávání dat mezi komponentami a jejich případnými vlákny, aniž by byla narušena zapouzdřenost, je zaveden systém událostí.

32 C++ podporuje vícenásobnou dědičnost a pro implementaci interface tak slouží čistě abstraktní třídy.

33 Za předpokladu použití patche *PREEMPT_RT*.

34 Z pohledu banky částic MCL, celkově v rámci celého modulu je to zanedbatelné.

5 IMPLEMENTACE ŘEŠENÍ

Vyvinutý lokalizační modul je náplní této práce a bude popsán v následujícím textu. Tvoří jej aplikace spouštěná spolu s ostatními procesy vozidla programem *vehicle_guardian*, viz. kapitola [2.3]. Aplikace je napsána v C++ a má cca 10 000 řádků organizovaných v 60ti souborech. Pro kompilaci postačuje kompilátor podporující normu C++98 a standardní Linuxové knihovny: *pthread.h* , *unistd.h*, *fcntl.h* , *termios.h* , *fcntl.h* , *sys/types.h* , *sys/stat.h* , *sys/shm.h*. V případě, že je dostupný kompilátor s podporou C++11, využijí se automaticky některé jeho knihovny[35], podobně je-li dostupná knihovna Boost[36]. Pokud potřebné knihovny chybí, použije se jejich minimální varianta, která je součástí kódové základny projektu.

Při vývoji byl kladen důraz na přenositelnost kódu a jeho znovu použitelnost. Velká část kódu je multiplatformní, nicméně některé komponenty pracující s operačním systémem nejsou a pro jejich nasazení na jiných platformách by bylo nutné je upravit[37]. Většina tříd je zcela zapouzdřena a byla testována samostatně.

5.1 ARCHITEKTURA APLIKACE

Aplikaci je možné rozdělit do 3 částí: zpracování senzorických dat (popř. dalších údajů z vozidla), subsystém map a samotné jádro lokalizace. Zjednodušený diagram je na [Obr. 15].

Pro predikci pohybu je využívána primárně odometrie, vypočítávána z dat poskytovaných inkrementálními senzory. Výstupem procesu zpracování odometrie je model pohybu, který je čten v predikční fázi výpočtu lokalizace jádrem MCL.

Každý zdroj lokalizace, ať už se jedná o senzor, WiFi, či odometrii, má vlastní třídu, která je k jádru MCL registrována. MCL v průběhu své činnosti tyto třídy volá s předanou částicí a třída na ni aplikuje model pohybu, či senzorický model. Jádro je tímto zcela odděleno od zbytku systému.

Modul dále obsahuje manager map, který se stará o nahrávání aktuálních mapových podkladů. Ty jsou následně využívány při generování nových vzorků, ale také při aplikaci modelů pohybu[38]. Některé třídy zpracovávající senzorická data vyžadují mapové podklady rovněž. Jedná se především o WiFi, kde je potřeba viditelné AP doplnit o jejich pozici, nebo v případě proximitních senzorů pro korelaci s mapou.

---

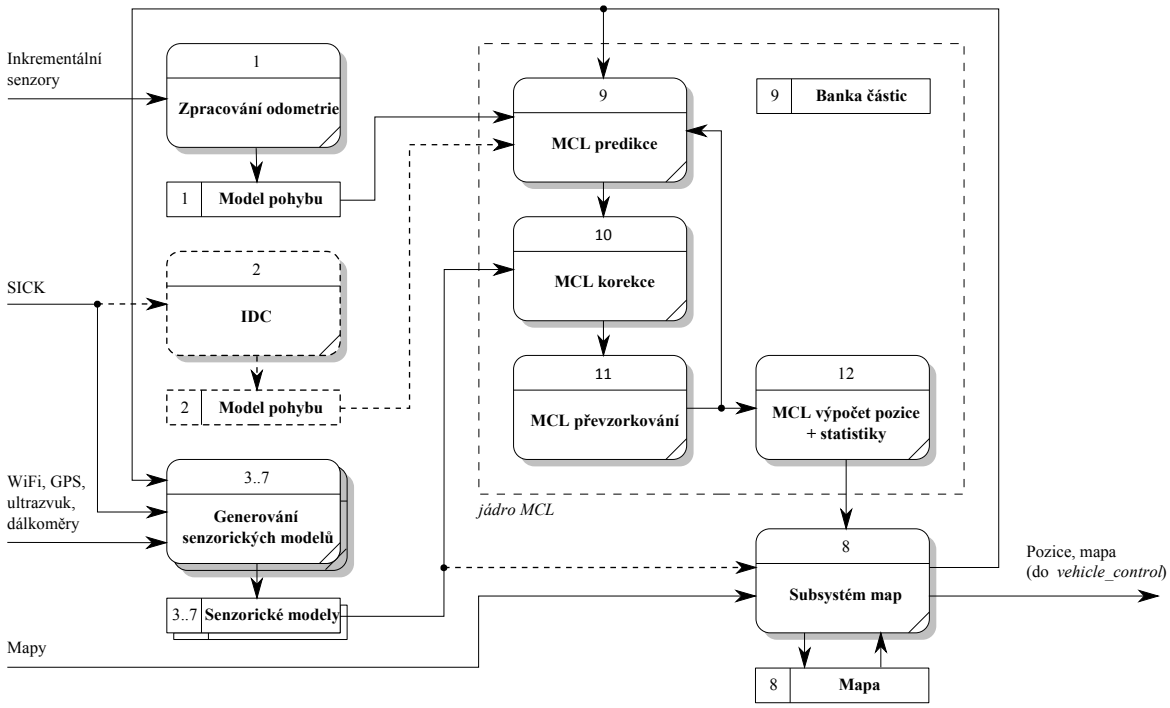
35 Kompilátor pro cílovou platformu nicméně podporuje pouze C++03 a experimentálně v minimálním rozsahu C++0x.

36 http://www.boost.org/.

37 Původně byli sice i tyto multiplatformní, ale s rostoucí kódovou základnou postrádalo smysl je takto nadále udržovat a v rámci refactoringu byly části specifické pro ne-Linuxové systémy odstraněny.

38 Vzorky se nesmí „pohybovat“ skrze zdi.

V případě laserového senzoru SICK LMS 100 je možné získaná data využít i pro souběžné mapování, nicméně toto bylo prozatím implementováno pouze jako simulace, viz. kapitola 6. Také je možné využít algoritmus Iterative Dual Correspondence[39] a výstupem doplnit model pohybu.



Obr. 15: Diagram datových toků modulu lokalizace

5.2 JÁDRO MCL

Jádro lokalizačního modulu tvoří banka částic, nad kterou operují třídy provádějící základní fáze MCL: translaci vzorků v predikčním fázi pomocí modelů pohybu, vážení vzorků podle dostupných senzorických modelů v korekční fázi, převzorkování a konečně samotný výpočet pozice metodou těžiště.

39 Kontinuální metoda založená na porovnávání sad nezpracovaných senzorických měření . Výstupem je translace a rotace aktuálního skenu k referenčnímu, získanému v minulém kroku.

O výpočet MCL se stará samostatné vlákno. Samotný algoritmus MCL je možné popsat následujícím pseudokódem:

```
1.  function mclAlgorithm(X_{t-1}, u_t, z_t, M)
2.      X_t = ∅
3.      for m = 1:M
4.          x_t^[m] = motionModel(u_t, x_{t-1}^[m])
5.          w_t^[m] = measurementModel(z_t, x_t^[m], m)
6.          X_t = X_t + ⟨x_t^[m]; w_t^[m]⟩
7.      end for
8.      X_t = resampling(X_t)
9.      return X_t
```

kde:

X_t	stavový prostor MCL; banka částic
u_t	řídící data (zde konkrétně data odometrie)
z_t	senzorická data
M	počet částic
$x_t^{[m]}$	stav částice *m* v čase *t* (pozice, směr)
$w_t^{[m]}$	*váha částice m v čase t*

5.2.1 Inicializace

Po zapnutí vozidla může před spuštěním nastat několik situací:

- Globální pozice je zadána explicitně operátorem.
- Globální pozice je známa, např. je viditelné některé AP s deskriptorem v mapě nebo je dostupná poloha z GPS. Přestože tyto hodnoty můžou být velmi nepřesné, je možné vybrat mapové podklady pro tuto oblast.
- Operátorem byla explicitně zadána oblast ve formě načtené mapy.
- Není známa žádná globální pozice ani nebyla explicitně zadána oblast. MCL v tomto případě vychází z poslední známé pozice.

Následně je načtena mapa oblasti ve které je lokalizace prováděna a jsou na ni rovnoměrně vygenerovány částice. V případě, že byla zadána konkrétní výchozí pozice operátorem, jsou všechny vzorky soustředěny do této pozice.

5.2.2 Generování částic

Pro generování částic se využívá generátor pseudonáhodných čísel s vhodným rozložením. Pro nové částice se používá rovnoměrné rozložení. Pro modelování nejistot modelů je použito normální, popř. trojúhelníkové rozložení.

Generátor náhodných čísel má vždy normální rozložení. Pro jiné rozložení existují v programu třídy, které zvolené rozložení generují z normálního.

1.2.2 Generátor pseudonáhodných čísel

Jedním ze zcela základních faktorů, ovlivňujících celou rodinu metod Monte Carlo je kvalita generátoru náhodných čísel. V jazyce C++ je k dispozici v <*stdlib.h*> funkce *rand*(). Přestože není implicitně normou definována implementace, bývá *rand*() typicky lineární kongruentní generátor (LCG) s periodicitou 2^{32} a na výstupu jsou dostupné bity 16 až 30[40]. V případě *glibc*[41] bývá *rand*() implementován kvalitnějším lineárním zpětnovazebním posuvným registrem (LFSR).

Ve specifikaci C++11 existuje knihovna <*random*>, která implementuje řadu komponent pro realizaci generátorů náhodných čísel a distribucí. Podobně i knihovna *BOOST*.

Jako generátor náhodných čísel použitý pro jádro lokalizace byl vybrán Mersenne Twister generátor pseudonáhodných čísel ve své 32bitové verzi, označovaný jako MT19937. Generátor má následující vlastnosti [22]:

- Generátor je 623 rozměrný s rovnoměrnou distribucí.
- Velmi dlouhá perioda 2^{19937}-1.
- Velmi vysoká rychlost generování čísel srovnatelná s LCG funkce *rand*().
- Projde většinou empirických statistických testů generátorů náhodných čísel s rovnoměrným rozložením TestU01[42].

Generátor je založen na maticové lineární rekurenci nad konečným bitovým polem. Matematický popis je možné nalézt v [22]. Pro uchování svého stavu používá pole 624 32bitových slov. Aby nebylo nutné vyžadovat pro tento generátor závislosti na externích knihovnách, byl MT19937 převzat z referenční implementace na [23] a upraven do generické šablony po vzoru *std::mersenne_twister_engine* z <*random*>. Implementace je zajímavá především téměř výhradním používáním rychlých bitových operací bez operací dělení, či násobení. Celkem náročné operace modulo jsou eliminovány rozdělením smyčky aktualizující stavový vektor, viz. implementace.

MT19937 musí být před použitím inicializován[43], protože při nenáhodném naplnění stavového vektoru (zejména při výskytu velkého množství nul) může generátor produkovat po mnoho iterací nekvalitní výstup[44]. Inicializace stavového vektoru je možné provést pomocí např. LCG, či LFSR,

---

40 Maximální hodnota výstupu je dána definicí *RAND_MAX*.

41 GNU C Library, standardní knihovna jazyka C používána v kombinaci s kompilátorem GCC typicky na svobodných OS s obrovskou podporou hardwarových platforem.

42 http://www.iro.umontreal.ca/~simardr/testu01/tu01.html.

43 Stavový vektor musí být naplněn tzv. *seed*em.

44 Ve smyslu statistických testů náhodnosti, například se ve výstupu může vyskytovat nějaký vzor v generovaných číslech.

které produkují čísla „s rozumnou náhodností“ brzy. Algoritmus kontroluje, zda nebyl stavový vektor naplněn pouze nulami, protože MT19937 se z tohoto stavu již nedokáže obnovit.

V Linuxu existují 2 speciální znaková zařízení poskytující rozhraní ke generátoru skutečně reálných čísel, který je součástí kernelu: */dev/random* a */dev/urandom*. Šum z okolí zachytávaný na periferiích a jiných zdrojů je sbírán do tzv. „entropy pool“, ze kterého jsou náhodná čísla generována. Pokud je entropy pool prázdný, respektive entropie nasbíraných dat je příliš nízká, zařízení */dev/random* zablokuje volající proces po dobu, dokud entropie nevzroste a náhodná data můžou být vrácena. Zařízení */dev/urandom* je neblokující. Pro využití jako hlavního generátoru algoritmu Monte Carlo však nejsou pro svou rychlost vhodné. Nicméně */dev/urandom* je použit pro inicializaci MT19937.

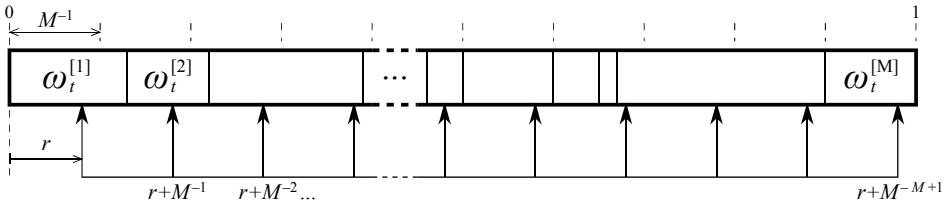
5.2.3 Převzorkování

Převzorkování hraje velmi důležitou roli a soustředí částice do pravděpodobných míst výskytu vozidla a tím i výpočetní výkon. Zjednodušeně řečeno nahrazuje původní sadu vážených vzorků sadou novou, kde mají všechny vzorky váhu 1 tak, aby bylo zachováno rozložení pravděpodobnosti.

V jádru MCL je implementováno převzorkování s nízkým rozptylem (Low variance resampling). Jeho implementaci shrnuje výpis:

```
1.  function lowVarinceSampler(X_t, W_t)
2.      X̄_t = ∅
3.      r = rand(0, M^-1)
4.      c = w_t^[1]
5.      i = 1
6.      for m = 1:M
7.          u = r + (m − 1) M^-1
8.          while u > c
9.              i = i + 1
10.             c = c + w_t^[i]
11.         end while
12.         X̄_t = X̄_t + ⟨x_t^[i]⟩
13.     end for
14.     return X̄_t
```

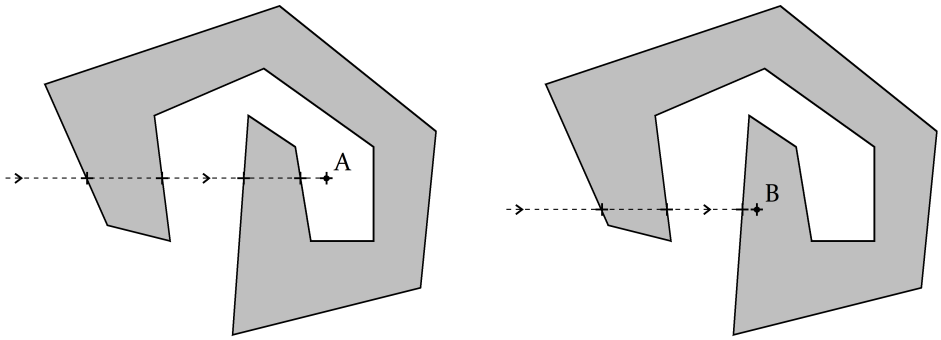
Nejdříve se vygeneruje náhodné číslo v rozsahu 0 až $1/M$, kde M je celkový počet částic. Počínaje tímto číslem se vygeneruje soustava M hodnot s krokem $1/M$. Částice jsou následně procházeny a jejich váha akumulována. Vždy, když číslo z vygenerované soustavy spadá do akumulované váhy částice, je tento vzorek přidán do výstupní banky částic. Situaci pro přehlednost znázorňuje [Obr. 16].



Obr. 16: Princip převzorkování s nízkým rozptylem

5.2.4 Integrace mapy do stavového prostoru

Nově generované vzorky (včetně generovaných převzorkováním), nebo jejich obrazy, na které byla aplikována predikce se můžou nově nacházet s jistou pravděpodobností kdekoliv ve stavovém prostoru. Pokud je k dispozici mapa prostoru a tedy je známo, ve kterých místech se nové vzorky nacházet nemůžou, můžeme tyto filtrovat.



Obr. 17: Detekce bodu uvnitř polygonu

Pro detekci kolize vzorku (bodu) s objektem v mapě[45] (polynomu) lze využít algoritmus trasování paprsků[46]. V této práci byl využit konkrétně derivát tohoto algoritmu PNPoly [24].

Algoritmus vychází z teorému Jordanovy křivky[47]. Z testovaného bodu vychází polopřímka a počítá se počet křížení s hranami polynomu. Pokud je počet křížení lichý, nachází se bod uvnitř polygonu, jinak mimo. Situaci ilustruje následující obrázek [Obr. 17].

45 Například zeď.

46 Ray casting algorithm.

47 Teorém Jordanovy křivky tvrdí, že jednoduchá uzavřená a nekřížící-se křivka rozděluje rovinu, na které leží na 2 oblasti

Algoritmus v pseudokódu je uveden v následujícím výpisu:

```
1.  function pnpoly(point, polygon)
2.      isInside = false;
3.      foreach line in polygon.lines
4.          if (line.start.y <= point.y <= line.end.y)
                    or (line.end.y <= point.y <= line.start.y)
5.              if point.x < line.start.x + (line.end.x - line.start.x)
                        *(point.y - line.start.y)/(line.end.y - line.start.y)
6.                  isInside = not(isInside)
7.              end if
8.          end if
9.      end for
10.     return isInside
```

5.3 PREDIKCE A MODELY POHYBU

Pro predikci pohybu je možné využít seznam akcí ze systému řízení. Protože jsou však na kolech nainstalovány inkrementální senzory, je použito mnohem přesnější odometrie.

Mimo odometrie je možné využít skenů z laserového senzoru SICK LMS 100 a algoritmu IDC, viz. zmínka v kapitole 5.1.

5.3.1 Odometrie

Data z inkrementálních senzorů na kolech jsou získávána ze sdílené paměti vozidla. Z počtu impulsů je počítána dráha na každém kole. Z těchto drah je možné vypočítat celkovou uraženou vzdálenost a úhel zatáčení vozidla následovně:

$$s_t = \frac{s_1 + s_2}{2} \tag{22}$$

$$\omega_t = \frac{s_1 - s_2}{d} \tag{23}$$

kde:

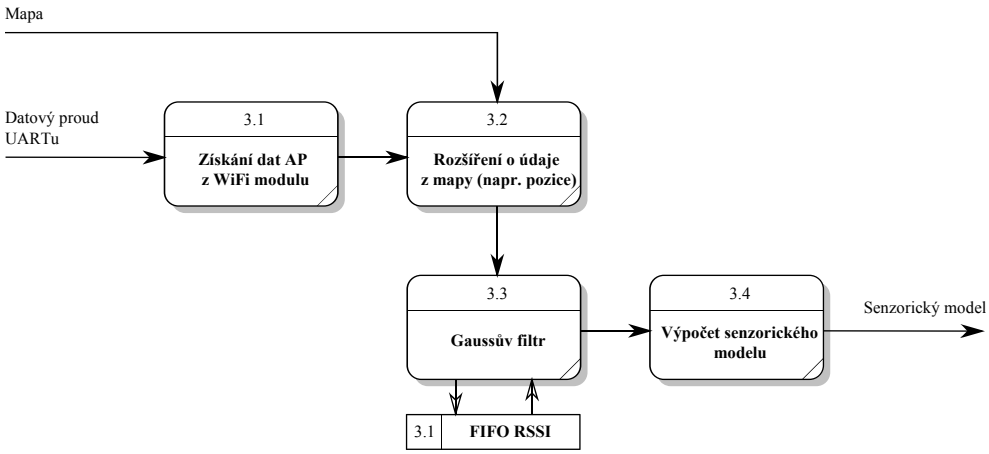
s_1, s_2	uražená dráha na kolech (m)
s_t	uražená dráha vozidla (m)
ω_t	zatočení vozidla (rad)
d	rozvor kol/podvozku (m), viz. kapitola 2.1.

5.4 KOREKCE A MODELY SENZORŮ

Pro korekci jsou využívány senzorická data dostupná ze sdílené paměti vozidla, data získaná komunikací s WiFi modulem a GPS.

5.4.1 WiFi

Zpracování dat WiFi má primárně 2 vstupy a 1 výstup. Vstupem je adresa portu, na kterém je připojen WiFi modul a načtená mapa (konkrétně pole deskriptorů AP). Výstupem pak pozice včetně tolerancí. Celý proces lokalizace znázorňuje diagram na [Obr. 18].



Obr. 18: Diagram datových toků zpracování AP WiFi

Po spuštění je otevřen port s WiFi modulem a spuštěno vlákno, které periodicky vyčítá informace o viditelných AP a tvoří z nich seznam párů MAC adresy a hodnoty RSSI.

Seznam párů je dále předán procesu, který tyto páry rozšíří o informace z mapy. Jedná se o souřadnice, zisk antény, ztrátu šířením a ztrátový koeficient. Pokud některá z těchto informací chybí, je použita defaultní hodnota. Páry, které nemají své údaje v mapě, jsou odstraněny.

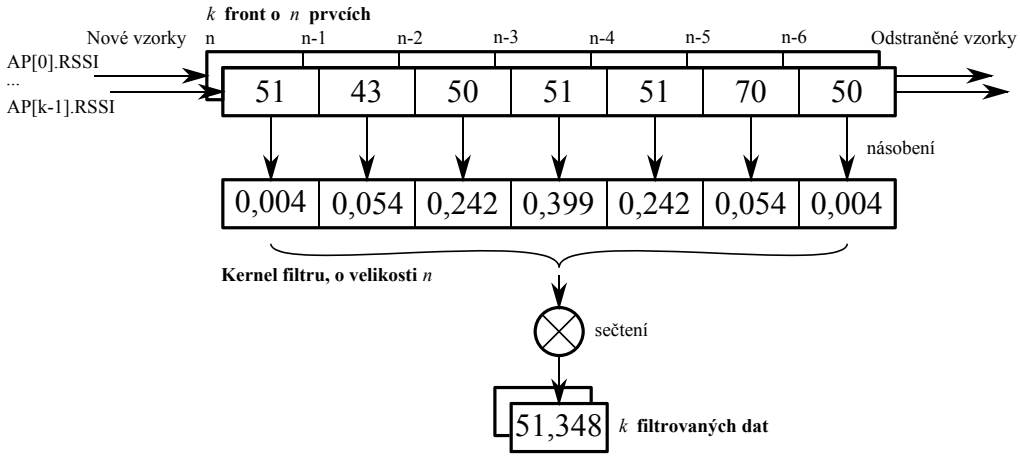
Do budoucna je možné mapy rozšířit i o další

Další fází je záznam RSSI do historie a Gaussova filtrace RSSI. Výsledkem tohoto kroku je seznam položek AP, které mají pozici, vyfiltrovanou hodnotu RSSI a další potřebné údaje k výpočtu vzdálenosti vozidla k jednotlivým AP.

Výpočet vzdáleností k jednotlivým přístupovým bodům WiFi je následně z filtrovaných RSSI počítán pomocí Path-Lost modelu a následně je z kombinace pozice a vzdálenosti vytvořen senzorický model. Nejistota vzdáleností získaná Path-Lost modelem je modelována normálním rozložením.

Rutina Gaussova filtru

Matematický popis Gaussova filtru byl uveden v kapitole 3.3.2. Princip funkce, tak jak je implementován, je zobrazen na [Obr. 19].



Obr. 19: Funkce Gaussova filtru

V rutině Gaussova filtru je založena mapa[48] front o velikosti shodné s velikostí kernelu. Klíče jednotlivých front jsou tvořeny MAC adresou filtrovaného RSSI páru. Algoritmus lze popsat následujícím pseudokódem:

```
1.    foreach ItemAP in ListAP
2.        if exist(FilterMap[ItemAP.MAC])
3.            FilterMap.add(ItemAP.MAC, new FilterQueue())
4.        end if
5.        FilterMap[ItemAP.MAC].push(ItemAP.RSSI)
6.    end foreach
7.    foreach FilterQueue in FilterMap
8.        if FilterQueue.size == Kernel.size + 1
9.            FilterQueue.pop()
10.       end if
11.       if FilterQueue.size == 0
12.           FilterMap.remove(FilterQueue)
13.       end if
14.   end foreach
```

V první fázi jsou ze vstupních párů AP vybírány MAC adresy. Pokud mapa obsahuje klíč s danou MAC adresou, je do příslušné fronty zařazena nová hodnota RSSI, jinak je fronta vytvořena a následně je zapsáno nové RSSI. Následně je ze všech front jeden prvek odebrán. V dalším kroku je ze všech front vytvořena matice a ta je vynásobena kernelem. Získaný vektor reprezentuje filtrovaná data všech viditelných AP. Tyto filtrovaná data jsou zpětně zapsána do párů a vrácena k dalšímu zpracování.

48 C++ šablona std::map - datový kontejner s položkami, které jsou indexovatelné podle klíče

Výpočet vzdáleností od přístupových bodů WiFi

Výpočet vzdálenosti je počítán podle vztahu (5). K výpočtu je nutné znát kromě samotné hodnoty RSSI ještě zisk antény přijímače, výkon vysílače, ztráty šířením a spádový koeficient, které jsou získávány z map.

Pan Ing. Ondřej Krejcar, Ph.D. ve své práci „Využití lokalizace uživatele pro prediktivní nahrávání dat v řídicích systémech“ vyvodil na základě měření tzv. superideální křivku závislosti RSSI na vzdálenosti od AP. Tato křivka je popsaná vztahem [14]:

$$y = 13{,}21 \ln(x) - 30{,}292 \tag{24}$$

kde: y relativní ztráty šířením (dBm)
x vzdálenost od vysílače (m)

Porovnáním s Path-Lost modelem můžeme získat následující koeficienty potřebné pro výpočet:

$l_0 = 30{,}292$ dBm ztráty šířením (dBm)
$n = 3{,}042$ spádový koeficient (-)

Tyto koeficienty budou použity jako defaultní pro AP, u kterých tyto informace nejsou v mapě zaznamenány.

Výpočet pozice na základě dat z WiFi modulu

Pro hrubý odhad pozice po zapnutí zařízení, v případech, kdy je pozice neznáma se využívá multilaterace, viz. kapitola [3.3.1]. Takto vypočtená pozice je použita pouze k počátečnímu načtení mapových podkladů dané oblasti a nejsou na ni tedy kladeny přísné požadavky, přestože dosahovaná přesnost může za jistých podmínek být lepší než 1m. Jedná se o systém vyvinutý v rámci BP [2].

Připojení WiFi modulu

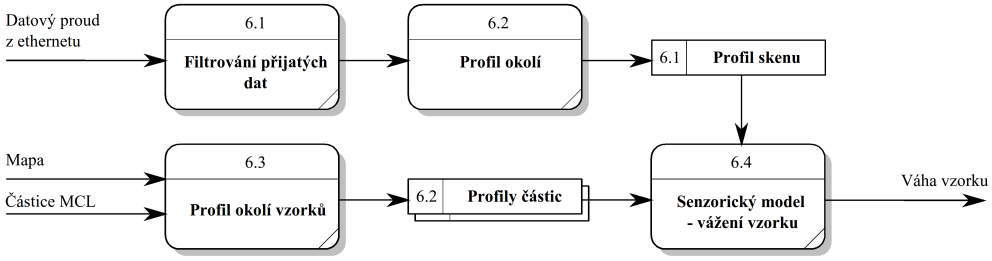
Pro skenování okolních AP je použit modul OWSPA311g od firmy Connect Blue . Protože na kitu již nejsou dostupné využitelné sériové porty, viz. kapitola [2], je modul připojen přes převodník USB<>UART od firmy FTDI připojený na USB. Protože kernel použitý původně ve vozidlu neobsahuje ovladače pro FTDI, které v systému vytvoří příslušné znakové zařízení, byl zkompilován s příslušnou konfigurací kernel ve verzi 2.6.24. Podrobný návod kompilace kernelu s podporou FTDI a řešením problémů s maximálním odebíraným proudem z USB je v příloze „Kompilace kernelu s podporou pro FTDI “.

5.4.2 GPS

Data z GPS dostává lokalizační modul ze sdílené paměti, včetně předpokládané přesnosti.

5.4.3 SICK

Laserový senzor SICK LMS 100 je připojen přímo k hlavní desce přes ethernet. Protože produkuje velké množství dat, jsou vybírány vzorky jen v určitých intervalech, aby se snížila výpočetní náročnost algoritmu.



Obr. 20: Diagram datových toků zpracování senzoru SICK

Přijatý sken reprezentuje profil okolí a je korelován s profilem zpracovávaného vzorku podle dostupné mapy. Výsledek této korelace pak ovlivňuje vážení vzorku. Diagram datových toků je na [Obr. 20].

5.4.4 Ultrazvuk a infračervené proximitní dálkoměry

Aktuálně nejsou pro lokalizaci využity, nicméně jejich implementace by byla principiálně obdobná se senzorem SICK.

5.5 SUBSYSTÉM MAP

Mapy jsou využívány při vytváření senzorických modelů proximitních senzorů, jako zdroj deskriptorů AP WiFi pro model WiFi a také jako filtr vzorků, které leží ve zdi. Mapy obsahují také definice cest, podle kterých vozidlo plánuje v autonomním režimu pomocí Dijkstrova algoritmu svou trasu. V aktuální implementaci jsou mapy pouze čteny a jsou vektorové.

Vozidlo podporuje 2 formáty mapových podkladů – soubory **.map* a **.evm.json*. Původní formát **.map* je pro potřeby lokalizace rozšířen o deskriptory AP WiFi a definice objektů modifikována. S ohledem na budoucí rozšiřitelnost byla přidána podpora nových mapových podkladů ve formátu **.evm.json*.

5.5.1 Datový formát MAP

Formát **.map* jsou jednoduché textové soubory. Mapa je rozdělena do čtvercových segmentů organizovaných v mřížce. Společně se soubory indexu segmentů **.mip* umožňuje tato organizace velmi rychlé hledání. Mapy používají kartézský souřadnicový systém s počátkem pozicovaným na absolutní zeměpisné souřadnice pomocí GPS souřadnic [3]. Ty tvoří zároveň hlavičku souboru[49]:

49 Některé příklady budou pro jistou názornost uváděny s konkrétními hodnotami.

```
1.  Loc: 49°49'29.367"N, 18°12'48.519"E
2.  Loc: 49°50'20.729"N, 18°13'8.952"E
```

První souřadnice definuje umístění levého dolního rohu mapy (počátku kartézského souřadnicového systému) a druhá pravého horního. Souřadnice jsou udávány v mm.

Po této hlavičce následují definice jednotlivých segmentů. Každý segment obsahuje svá metadata – ID segmentu, svou pozici v kartézských souřadnicích, velikost a reference na ID sousedních segmentů. Sousední segmenty se počítají od levého horního ve směru hodinových ručiček. ID 0 znamená, že neexistuje v daném směru sousední segment a aktuální segment je tedy hraniční. Segmenty jsou uvozeny klíčovým slovem *STARTSEGMENT* a ukončeny slovem *KONECSEGMENT*. Za metadaty do slova *KONECSEGMENT* jsou umístěny samotné objekty mapy.**STARTSEGMENT**

```
3.  ID=1
4.  X0=0
5.  Y0=0
6.  SIZE=50000
7.  SEGMENTREF1 0
8.  SEGMENTREF2 0
9.  SEGMENTREF3 3
10. SEGMENTREF4 2
11. SEGMENTREF5 0
12. SEGMENTREF6 0
13. SEGMENTREF7 0
14. SEGMENTREF8 0
15. … //objekty segmentu
16. KONECSEGMENT
```

Objekty jsou definovány 3: cesty, polygony (jako např. zdi) a deskriptory AP WiFi. Původní soubory map definovali zdi jako sadu úseček, které navíc nemuseli následovat po sobě. To by byl vcelku výrazný problém, protože např. při testování, zda částice MCL neleží ve zdi by bylo zapotřebí nejdříve nalézt všechny úsečky, které tvoří polygon zdi a to případně i přes několik sousedních segmentů. Tato operace by byla časově velmi náročná[50] a z tohoto důvodu byl popis úsečkami nahrazen polygony. Definice objektu polygonu vypadá následovně:

```
1.  <vrstva> <vertex[0].x> <vertex[0].y> <vertex[1].x> <vertex[1].y> …
    … <vertex[n=1].x> <vertex[n-1].y>
```

kde:	<*vrstva*>	typ polynomu, může nabývat hodnot „*ZED*“, „*DVERE*“, nebo „*CESTA*“
	<*vertex[i].x*>	$i \in \{0; 1; \ldots; n\text{-}1\}$, X-ová souřadnice *i*-tého vertexu
	<*vertex[i].y*>	$i \in \{0; 1; \ldots; n\text{-}1\}$, Y-ová souřadnice *i*-tého vertexu

50 Pro všechny částice by bylo nutné nalézt všechny úsečky náležící konkrétnímu polygonu, který kříží aktuální segment částice.

Zvláštním objektem jsou deskriptory AP definované následovně:

```
1.  AP <x> <y> <mac> <gain> <path_lost> <propagation_exponent> <ssid>
```

kde:	<*x*>	X-ová souřadnice AP
	<*y*>	Y-ová souřadnice AP
	<*mac*>	MAC adresa ve formátu „xx:xx:xx:xx:xx:xx“
	<*gain*>	zisk na straně antény v dBm
	<*path_lost*>	ztráta šířením v dBm
	<*propagation_exponent*>	ztrátový koeficient
	<*ssid*>	SSID AP, zatím není využito

Soubor map je po všech definicích souborů ukončen klíčovým slovem *ENDOFFILE*.

Soubor indexu segmentů

K souborům *.map můžou být dodávány soubory indexu segmentů **.mip*. Jedná se o textové soubory se záznamy [3]:

```
1.  <id> <addr> <x> <y> <size>
```

kde:	<*id*>	ID segmentu
	<*addr*>	pozice v souboru **.map* obsahující segment s ID=<*id*>, konkrétně pozice 1. znaku klíčového slova *STARTSEGMENT*.
	<*x*>	X-ová souřadnice segmentu (shodná s X0=<*x*>)
	<*y*>	X-ová souřadnice segmentu (shodná s Y0=<*y*>)
	<*size*>	velikost segmentu (shodný s SIZE=<*size*>)

Index umožňuje velmi rychlé hledání segmentů, které je potřeba načíst do operační paměti, aniž by bylo nutné procházet segmenty v souboru **.map*. Výhody indexu se uplatní ve velmi velkých mapách.

Soubory **.mip* jsou již zastaralé a jsou nahrazeny binárním formátem generovaným automaticky při detekci změny mapových podkladů[51]. Není tedy třeba je generovat předem a distribuovat spolu s **.map*.

51 Používá se časového razítka modifikace souborů.

5.5.2 Datový formát EVM a parser JSON

Kromě formátu **.map* byla do vozidla přidána podpora formátu **.evm.json*. Opět se jedná o textový formát založený na notaci JSON. Tento formát byl přidán s ohledem na jeho čistou syntaxi, vysokou informační hustotou[52], snadnou a rychlou parsovatelnost, dobrou čitelnost i pro člověka, širokou podporu ve formě knihoven pro různé programovací jazyky a s tím související jednoduchost při exportech a konverzích. Oproti **.map* má ještě jednu výhodu – rozšiřitelnost. Modifikace, či doplnění formátu znamenají minimální zásah do softwarového vybavení a při rozumných úpravách také víceméně automatickou zpětnou kompatibilitu.

Příklad mapy obsahující jediný deskriptor WiFi AP a jeden polynom reprezentující zeď je na následujícím výpisu:

```
1.  {
2.      "filetype": "evm.json",
3.      "version": "1.00",
4.      "gps": "49°49'26.367\"N, 18°12'48.519\"E",
5.      "features": {
6.          "ap": [{
7.                  "mac": "00:23:F8:98:06:2A",
8.                  "gain": 50,
9.                  "pathlost": null,
10.                 "propagation": null,
11.                 "ssid": "AP002",
12.                 "coordinates": [1.3, 15.5]
13.             }
14.         ],
15.         "polygons": [{
16.                 "layer": "wall",
17.                 "coordinates": [
18.                     [3.4, 10.5],
19.                     [5.1, 10.5],
20.                     [5.1, 20.5],
21.                     [3.4, 20.5]
22.                 ]
23.             }
24.         ]
25.     }
26. }
```

52 JSON má nízký overhead – řídící značky tvoří malou část dat.

Výpis je celkem samo popisný, většina vlastností je shodná s popisem souborů *.*map*. Formát využívá opět kartézský systém. Souřadnice jsou v metrech. GPS souřadnice je pouze jediná a udává pozici počátku kartézského souřadnicového systému. Mapovací subsystém dokáže zpracovat několik formátů[53] (doporučován je však pouze první tvar)[54]:

- 49°49'26.367"N, 18°12'48.519"E
- 49,823990833N, 18,2134775E
- 18:12.80865'E, -49°49'26.367"S
- 49,823990833, 18,2134775

Kódová stránka musí být UTF-8, nicméně parser zpracuje všechny ASCII kompatibilní znakové sady.

Podrobný popis struktury mapy, včetně významu, možnosti některé položky vynechat a platných hodnot deklaruje JSON schéma v příloze A.

Parser JSON

Parserů formátu JSON existuje mnoho, nicméně pro potřeby vozidla je primární rychlost a z toho vycházející požadavek na sekvenční čtení i zpracování. Protože nebyla nalezena vhodná minimalistická knihovna splňující tento požadavek, řešení implementuje parser vlastní. Od jiných parserů se liší především:

- Sekvenční zpracování po položkách – tokenech.
- Vysoká rychlost čtení – cca 40 $MB \cdot s^{-1}$ (na notebooku Sony VAIO VGN-FW11E).
- Parsování přečtené hodnoty až na požádání, defaultně se vrací řetězec.
- Minimalistická kódová základna – cca 500 řádků.

Práci s parserem demonstruje následující výpis, který najde v mapě zisk na straně antény 1. WiFi AP a uloží jej do proměnné.

```
1.  IO::Files::JsonParser json;
2.  json.Open("mapfile.evm.json");
3.  double gain = 0;
4.  while (json.ReadNext()) {
5.      if (json.GetPath().Compare("features/ap/0/gain")) {
6.          gain = json.GetValue<double>();
7.          break;
8.      }
9.  }
```

53 Všechny formáty které je možné zpracovat definuje regulární výraz v JSON schématu v příloze A, popř. zdrojový kód GeographicCoordinates.cpp.

54 Všechny reprezentují stejnou polohu.

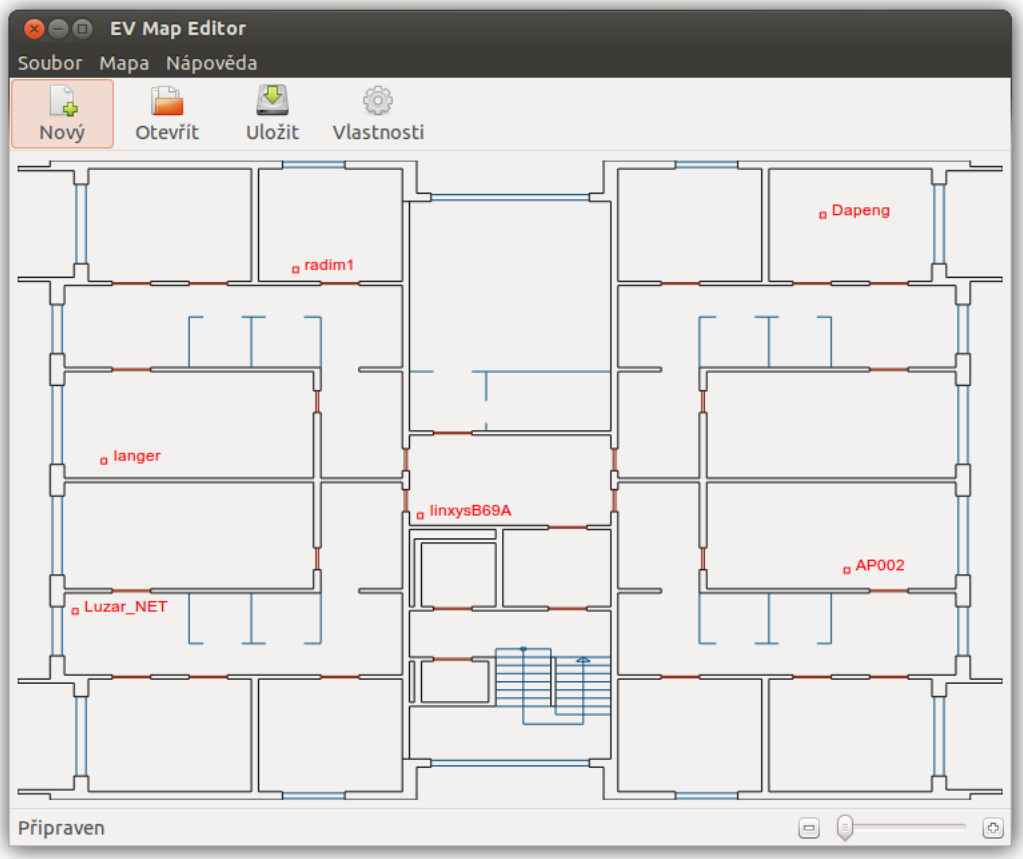
Parsování celé mapy probíhá v jediném průchodu a rychlost načtení mapy je tak velmi dobrá a přibližuje se rychlosti čtení souborů *.map. Načtení 1 objektu z mapy a jeho uložení trvá cca 60 ns.

1.2.3 Konverze a vytváření map

Mapy je vhodné kreslit v některém CAD editoru, který umožňuje export výkresu do formátu **.dxf* (ve verzi 12, nebo kompatibilní). Je možné použít například AutoCAD, FreeCAD.

Pro konverzi formátu **.dxf* do formátu, který používá vozidlo je připravena aplikace „EV map editor“ [Obr. 21], která vznikla přepsáním aplikace „Generátor map“ [3] od Ing. Jaromíra Konečného. Aplikace je napsána v C#, je multiplatformní a umožňuje následující operace s mapami:

- Podporované formáty: **.map*, **.evm.json* a **.dxf* (pouze import).
- Nastavení GPS souřadnice počátku.
- Editaci deskriptorů WiFi AP a jednotlivých objektů mapy.



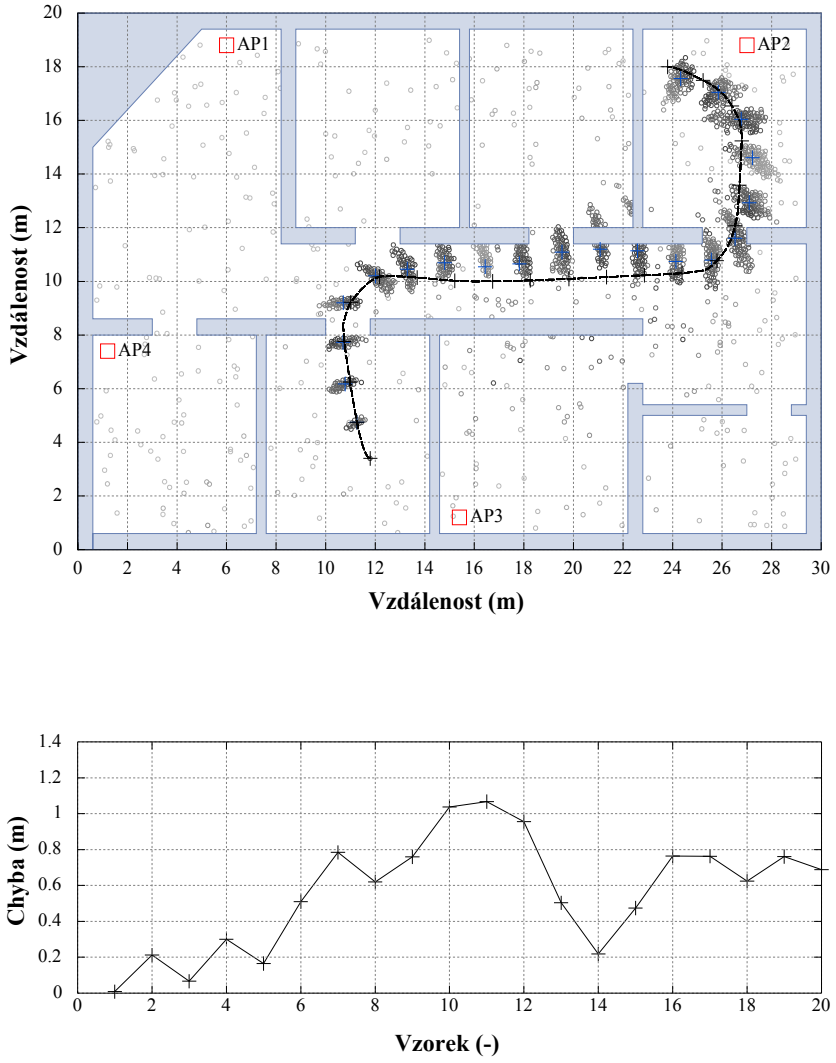
Obr. 21: Aplikace pro editování map vozidla

6 TESTOVÁNÍ

Pro ověření funkce lokalizačního modulu byla vytvořena sada testů a aplikována na zkompilovaný modul. Všechny testy jsou dostupné v přílohách a čítají cca 70 snímků a několik animací. Zde budou uvedeny nejvíce zástupné snímky.

6.1 LOKALIZACE S ZNÁMOU POČÁTEČNÍ POZICÍ

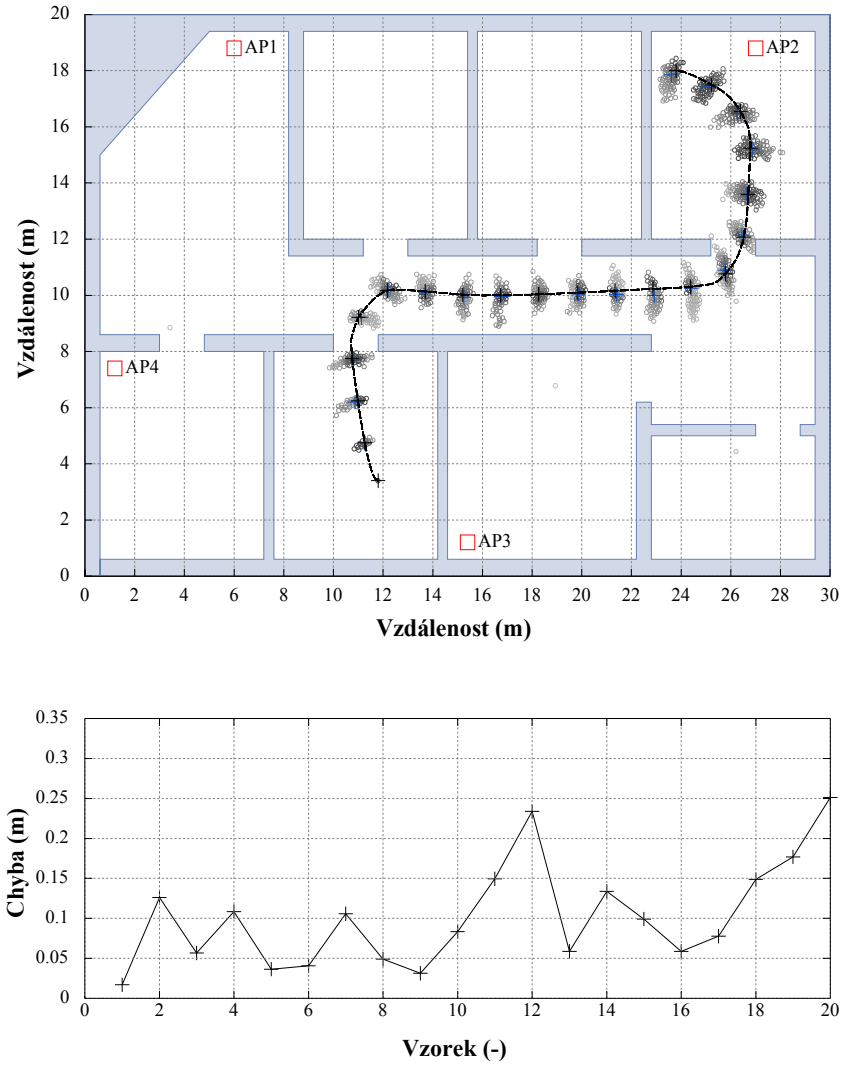
Scénář, kdy vozidlo ze známého výchozího bodu jede s využitím odometrie a WiFi:



Obr. 22: Lokalizace s odometrií a WiFi, 300 částic

Ze výše uvedených obrázků je zřejmé, že WiFi účinně koriguje chyby odometrie, i když sama je vcelku nepřesná. Z předchozích měření bylo zjištěno [2], že WiFi lokalizace může dosahovat přesnosti jednotek metrů, ve výjimečných případech s hustou sítí AP i méně, než 1 metr.

Následující scénář ukazuje stejný případ, kdy k WiFi byl přidán ještě senzor SICK, viz. [Obr. 23]. Ze senzoru jsou vybírány pouze údaje v 30° intervalech.

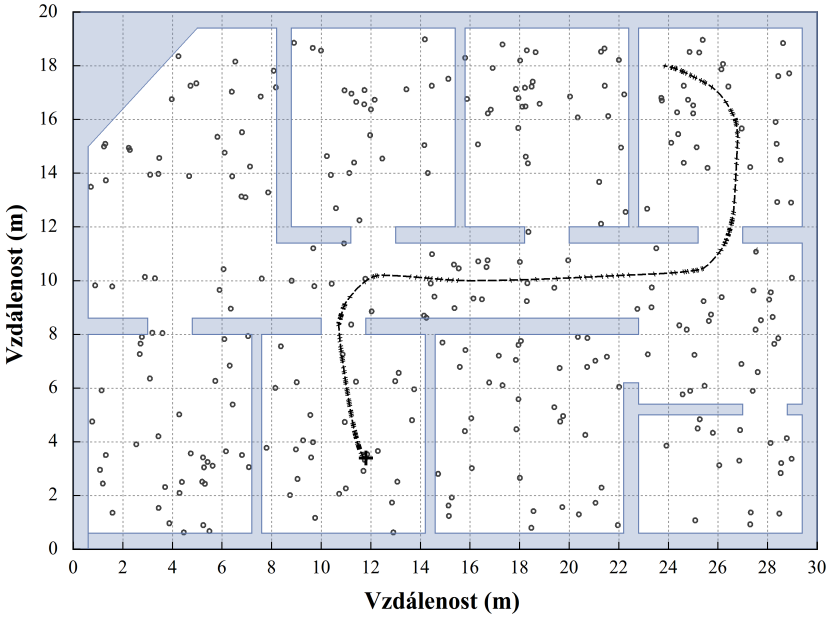


Obr. 23: Lokalizace s odometrií, WiFi a senzorem SICK, 300 částic

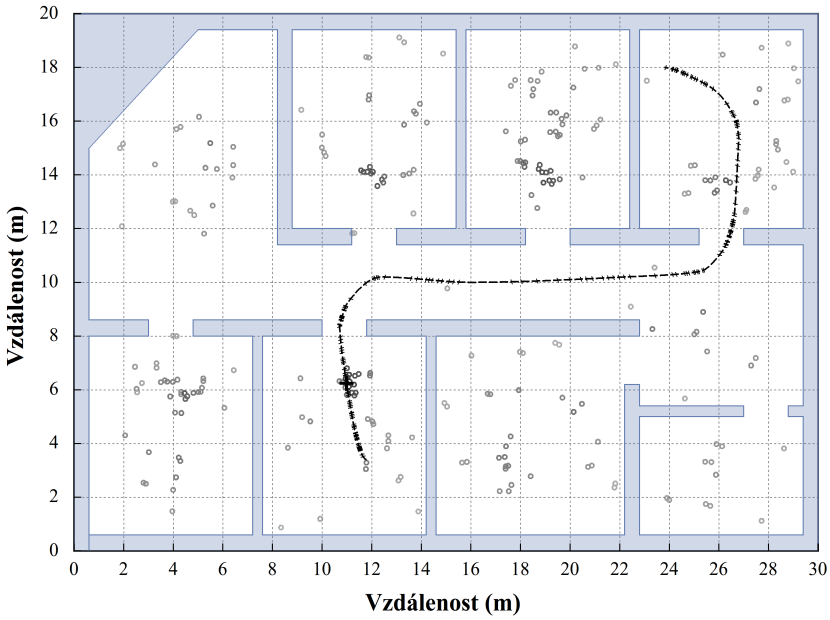
Průběh chyby je v tomto případě mnohem menší a pohybuje se v okolí 10 cm. Je tedy zřejmé, že senzor SICK dokáže velmi zpřesnit lokalizaci i s velkým intervalem vzorků.

6.2 GLOBÁLNÍ LOKALIZACE

Následující scénář ukazuje globální lokalizaci s využitím senzoru SICK, kdy neexistuje apriorní znalost výchozí pozice.

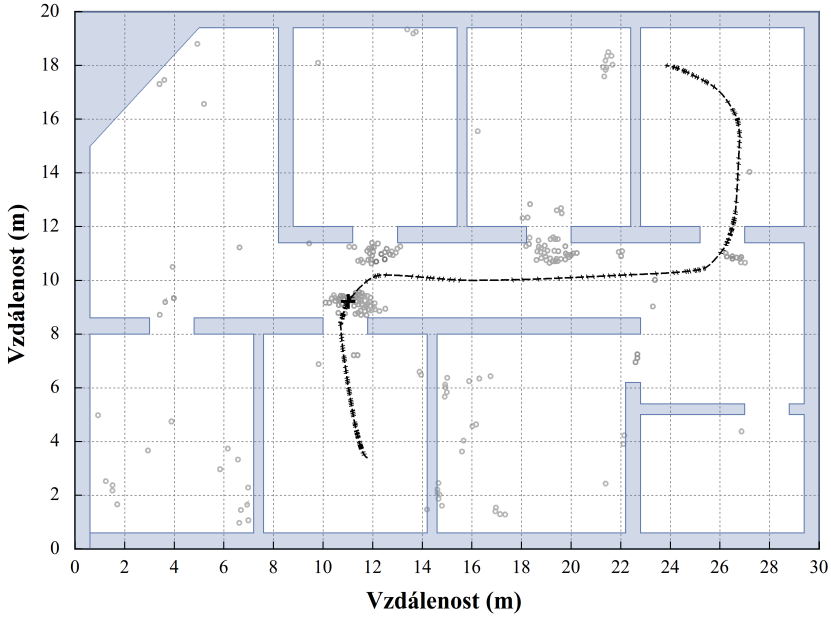


Obr. 24: Globální lokalizace se senzorem SICK, 0. iterace, 300 částic

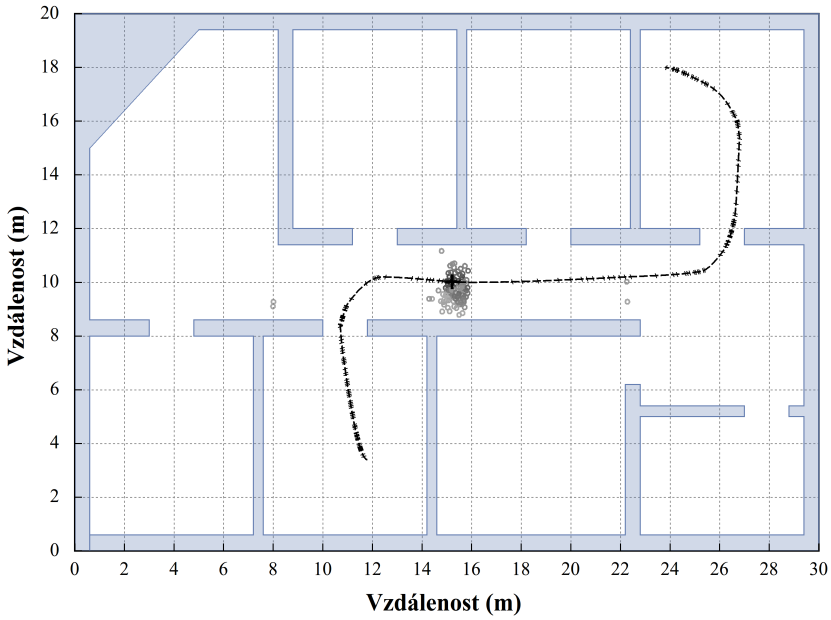


Obr. 25: Globální lokalizace se senzorem SICK, 2. iterace, 300 částic

Na [Obr. 25] je patrné, že prostředí je částečně symetrické a pravděpodobná pozice se soustředí na skutečnou pozici a její symetrické „obrazy“.



Obr. 26: Globální lokalizace se senzorem SICK, 4. iterace

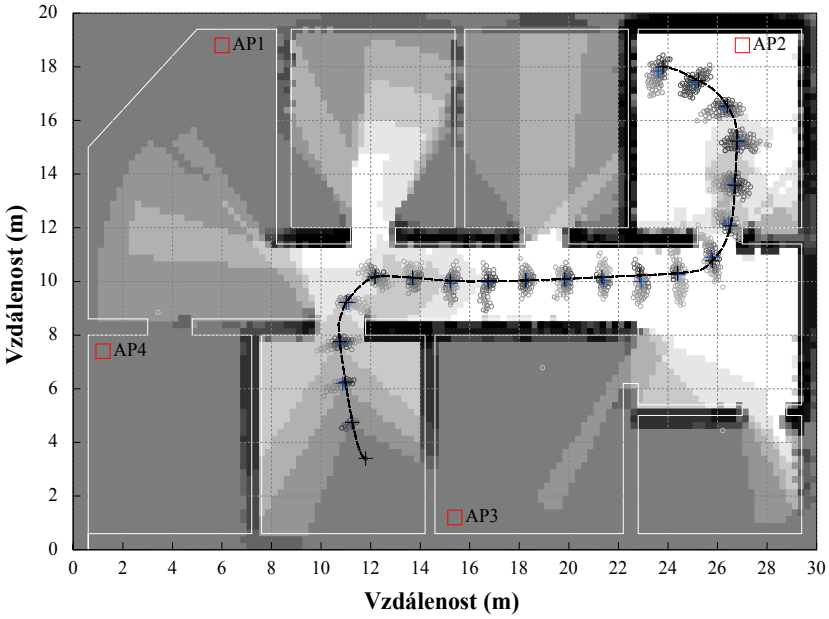


Obr. 27: Globální lokalizace se senzorem SICK, 7. iterace

Z [Obr. 27] je zřejmé, že vozidlo bylo zcela globálně lokalizováno v 7. kroku.

6.3 MAPOVÁNÍ SE SENZOREM SICK

Tato situace byla modelována v programu MATLAB/Octave a demonstruje mapování na mřížce během pohybu vozidla. V modulu lokalizace pro vozidlo toto ještě není implementováno. Nicméně z [Obr. 28] je zřejmé, co MCL z SICKu během lokalizace „vidí“.



Obr. 28: Mapování během lokalizace pomocí scanů ze SICKu

6.4 RYCHLOST GENEROVÁNÍ PSEUDONÁHODNÝCH ČÍSEL

Pro ověření rychlosti použitého generátoru pseudonáhodných čísel bylo provedeno měření v těsné smyčce:

Generátor	**počet iterací (-)**	**celková doba (s)**	**průměr (ns)**	**poměr (%)**
/dev/random	50	78,25	$1{,}57 \cdot 10^{6}$	0,00
/dev/urandom	10^{5}	0,186	1,861	1,08
MT19937	10^{7}	0,252	0,025	79,86
GCC rand()	10^{7}	0,202	0,020	100,00

Výsledek pro generátor Mersenne Twister je přívětivý a dosahuje cca 80 % rychlosti funkce rand() z *glibc*.

7 ZÁVĚR

Základní problém spojený s autonomními roboty je jejich jednoznačná lokalizace v prostoru. Lokalizace Monte-Carlo, která je implementována v rámci této práce přináší robustní a dále rozšiřitelný systém. Z testování vyplývá, že MCL může dosahovat vysoké přesnosti v řádu desítek centimetrů při využití laserového senzoru vzdáleností. Zároveň tato metoda umožňuje kromě kontinuální lokalizace také globální lokalizaci.

Zdrojové kódy obsahují mnoho zapouzdřených tříd organizovaných do jmenných prostorů, od šablony emulující události[55], tak jak jsou známé ze C#, přes knihovnu pro práci s maticemi až po debugovací nástroje. Zdrojové kódy tvoří dobrý základ pro další rozšíření, ale i pro zcela nové projekty.

Implementované řešení je funkční a bylo ověřeno testováním, simulacemi, na vývojovém kitu s připojeným WiFi modulem spolu s ostatními procesy vozidla, které je možné provozovat samostatně (*vehicle_init, vehicle_guardian, vehicle_memory*). Ve vozidle aktuálně řešení prozatím implementováno z časových důvodů a po domluvě s vedoucím práce nebylo. Integrace do stávajícího systému je tedy splněna pouze částečně a bude náplní následujících dnů.

Logickým pokračováním práce je kontinuální mapování, aktivní lokalizace a další zdokonalení plánovacích algoritmů autonomního řízení. První krok k mapování byl uskutečněn v rámci testování jako model v MATLABu/Octave. Dokud nebude kontinuální lokalizace hotova, je možné využívat program *EV Map Editor* pro konverzi výkresů z např. AudoCADu, čí FreeCADu. Pro měření parametrů se může hodit také program *WiFi Locator*, který vznikl v rámci bakalářské práce [2].

55 Jedná se o tzv, syntaktický cukr (Syntactic sugar).

8 POUŽITÁ LITERATURA

[1] *Garmin* [online]. 2011 [cit. 2012-12-17]. GARMIN | What is GPS?. Dostupné online: <http://www8.garmin.com/aboutGPS/>.

[2] KUREČKA , A. *Vývoj embedded modulu pro podporu lokalizace průzkumného vozidla*. Ostrava, 2011. 58 s. Bakalářská práce. Vysoká škola báňská - Technická univerzita.

[3] KONEČNÝ, J. *Hlavní řídicí jednotka pro mobilní robotické zařízení*. Ostrava, 2010. 75 s. Diplomová práce. Vysoká škola báňská - Technická univerzita.

[4] ANDERSON, J. *I.MX LITEKIT*. Washington: Logic PN, 2008. 28 s.

[5] Laser Measurement Sensors of the LMS1xx Product Family , Operating instructions. Germany: SICK, 2012. 96 s. Dostupné online: <https://www.mysick.com/saqqara/get.aspx?id=im0031331>.

[6] *Getting Started with Linux on the Logic PD i.MX31 Lite Kit*. Pittsburgh: TimeSys Corporation, 2007. 53 s.

[7] LIPKA , T. *Inovace vzdáleného ovládacího panelu pro mobilní robotické zařízení*. Ostrava, 2011. 48 s. Bakalářská práce. Vysoká škola báňská - Technická univerzita.

[8] LIPPA , T. *Vzdálený řídicí systém mobilního robotického zařízení*. Ostrava, 2010. 81 s. Diplomová práce. Vysoká škola báňská - Technická univerzita.

[9] LOGIC PD. *i.MX31 SOM-LV Product Brief*. Washington: [s.n.], 2009. 2 s.

[10] *Embedded Linux From A Trusted Source | Timesys Embedded Linux* [online]. 2011 [cit. 2012-12-17]. EMBEDDED Linux for Logic PD SOMs | Timesys Embedded Linux. Dostupné online: <http://www.timesys.com/supported/boards/logic>.

[11] KONEČNÝ, J; PROKOP, H; LIPPA, T. *I.MX31 Litekit – Postup instalace a oživení*. Ostrava: Vysoká škola báňská - Technická univerzita, 2009. 48 s.

[12] ADDESSO, P; BRUNO, L; RESTAINO, R. Adaptive localization techniques in WiFi environments. In . ISWPC. New York : IEEE Press, 2010. s. 289-294. ISBN 978-1-4244-6855-3 , DOI:10.1109/ISWPC.2010.5483731.

[13] VELJO, Otsason, et al. Accurate GSM Indoor Localization. In *7th International Conference on Ubiquitous Computing*. Amsterdam: Elsevier Science Publishers, 2005. s. 141-158. DOI: 3-540-28760-4.

[14] KREJCAR, O. *Využití lokalizace uživatele pro prediktivní nahrávání dat v řídicích systémech*. Ostrava, 2008. 166 s. Dizertační práce. Vysoká škola báňská - Technická univerzita.

[15] YUAN, Zhang, et al. Localization Algorithms for Wireless Sensor Retrieval. *The Computer Journal*. 2010, 53, s. 1594-1605. ISSN 0010-4620.

[16] YUNHAO, L; ZHENG, Y. *Location, Localization and Localizability: Location-awareness Technology for Wireless Networks*. 1. New York : Springer, 2011. 154 s. ISBN 978-1-4419-7370-2, DOI 10.1007/978-1-4419-7371-9.

[17] Wikipedie, otevřená encyklopedie [online]. 2011 [cit. 2013-04-20]. *Metoda nejmenších čtverců*. Dostupné online: <http://cs.wikipedia.org/wiki/Metoda_nejmenších_čtverců>.

[18] Wikipedia, the free encyclopedia [online]. 2009 [cit. 2013-04-20]. *Gaussian filter*. Dostupné online: <http://en.wikipedia.org/wiki/Gaussian_filter>.

[19] KALMAN, R. E. *A new approach to linear filtering and prediction pro-blems*. Transactions of the ASME–Journal of Basic Engineering, sv. 82, s. 35–45, 1960.

[20] Vybraná témata z mobilní robotiky. Seminář na VUT Brno 17. 2. 2011. RNDr. Miroslav Kulich Ph.D. a Dr.rer.nat. Martin Saska.

[21] THRUN, S; BURGARD, W; FOX, D. *Probabilistic robotics*, The MIT Press, 2005. 668 s. ISBN 978-0-2622-0162-9

[22] MATSUMOTO, M; NISHIMURA, T. *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.* ACM Transactions on Modeling and Computer Simulatio – TOMACS Journal, sv. 8, s. 3–30, 1998.

[23] Department of Mathematics, Hiroshima University [online]. 2011 [cit. 2013-05-02]. *Mersenne Twister Home Page*. Dostupné online: <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>

[24] RANDOLPH FRANKLIN, W. R. *PNPOLY - Point Inclusion in Polygon Test [online]. 2009 [cit. 2013-05-05].* Dostupné online: <http://www.ecse.rpi.edu/~wrf/Research/Short_Notes/pnpoly.html>

9 SEZNAM PŘÍLOH

Všechny přílohy jsou umístěny na přiloženém CD. CD obsahuje:

A. Vlastní práci v elektronické podobě (doc/dp-kur219.pdf)
B. Zdrojové kódy lokalizačního modulu (složka app/EVWorkspace)
C. Zdrojové kódy editoru map (složka app/EVMapEditor)
D. Data měření a simulací (složka measurements)
E. Návod „Kompilace kernelu s podporou pro FTDI “ (attachements/kernel-compile.pdf)
F. Použitou literaturu v elektronické podobě (složka e-books)

A. Schéma formátu EVM

Následující výpis tvoří specifikaci formátu mapových podkladů EVM definovanou schématem JSON, viz. http://json-schema.org/latest/json-schema-core.html.

```
1.  {
2.      "$schema": "http://json-schema.org/draft-04/schema#",
3.      "title": "EV Map",
4.      "description": "Mapovy podklad vozidla EV.",
5.      "type": "object",
6.      "required": true,
7.      "properties": {
8.          "filetype": {
9.              "description": "Typ souboru; musi byt \"evm.json\".",
10.             "type": "string",
11.             "enum": ["evm.json"],
12.             "required": true
13.         },
14.         "version": {
15.             "description": "Verze formatu.",
16.             "type": "string",
17.             "pattern": "/[1-9]\\.[0-9]{2}/",
18.             "required": false
19.         },
20.         "gps": {
21.             "description": "GPS souradnice pocatku mapy.",
22.             "type": "string",
23.             "pattern": "/([^a-zA-Z0-9+-]*[0-9]+(\\.[0-9]+)?){1,3}
    ([^SNWE]*[SNWE])?[^,]*,(([^a-zA-Z0-9+-]*[0-9]+(\\.[0-9]+)?){1,3}
    ([^SNWE]*[SNWE])?)?\\.*/",
24.             "required": true
25.         },
26.         "features": {
27.             "description": "Objekty mapy.",
28.             "type": "object",
29.             "required": false,
30.             "properties": {
31.                 "ap": {
32.                     "description": "Seznam pristupovych bodu WiFi.",
33.                     "type": "array",
34.                     "required": false,
35.                     "items": {
36.                         "description": "Pristupove body WiFi.",
37.                         "type": "object",
38.                         "required": false,
39.                         "properties": {
```

```
                            "mac": {
                                "description": "MAC adresa WiFi AP.",
                                "type": "string",
                                "pattern": "/([a-zA-Z0-9]{2}\\.){5}[a-zA-Z0-9]{2}/",
                                "required": true
                            },
                            "gain": {
                                "description": "Zisk na antene WiFi AP v dBm. Pokud neni uveden, nebo je null, pouzije se defaultni hodnota.",
                                "type": ["number","null"],
                                "minimum": 0,
                                "exclusiveMinimum": true,
                                "required": false
                            },
                            "pathlost": {
                                "description": "Koeficient ztraty sirenim signalu v dBm. Pokud neni uveden, nebo je null, pouzije se defaultni hodnota.",
                                "type": ["number","null"],
                                "minimum": 0,
                                "exclusiveMinimum": true,
                                "required": false
                            },
                            "propagation": {
                                "description": "Spadovy koeficient. Pokud neni uveden, nebo je null, pouzije se defaultni hodnota.",
                                "type": ["number","null"],
                                "minimum": 0,
                                "exclusiveMinimum": true,
                                "required": false
                            },
                            "ssid": {
                                "description": "SSID retezec WiFi AP. Aktualne neni vyuzit a ma spise informacni charakter.",
                                "type": "string",
                                "required": false
                            },
                            "coordinates": {
                                "description": "Kartezske souradnice AP ve tvaru [X,Y,Z]. Aktualne neni souradnice Z vyuzita a nemusi byt uvedena.",
                                "type": "array",
                                "minitems": 2,
                                "required": true,
                                "items": {
                                    "description": "Souradnice AP v dane ose.",
                                    "type": "number",
                                    "required": true
```

```
                                }
                            }
                        }
                    }
                },
                "polygons": {
                    "description": "Seznam polynomu objektu mapy.",
                    "type": "array",
                    "required": false,
                    "items": {
                        "description": "Polynomy objektu mapy.",
                        "type": "object",
                        "required": false,
                        "properties": {
                            "layer": {
                                "description": "Vrstva objektu. Aktualne je zpracovavana pouze vrstva \"wall\".",
                                "type": "string",
                                "enum": ["wall", "door", "way"],
                                "required": true
                            },
                            "coordinates": {
                                "description": "Vertexy polynomu objektu.",
                                "type": "array",
                                "minitems": 2,
                                "required": true,
                                "items": {
                                    "description": "Kartezske souradnice vertexu ve tvaru [X,Y,Z]. Aktualne neni souradnice Z vyuzita a nemusi byt uvedena.",
                                    "type": "array",
                                    "minitems": 2,
                                    "required": true,
                                    "items": {
                                        "description": "Souradnice vertexu v dane ose.",
                                        "type": "number",
                                        "required": true
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

```
124.  }
```